ROBUST ESTIMATION OF STATISTICAL TEMPORAL NETWORKS FOR FINANCIAL TIME SERIES MODELING: THEORETICAL FORMULATION AND TEMPORAL PATTERNS TYPOLOGY

MPhil Thesis

by Stilianos Pantelidakis

First Supervisor: Dr Nikolay NIKOLAEV

Second Supervisor: Prof. Robert ZIMMER

Department of Computing

Goldsmiths College

University of London

London SE14 6NW

18 / 2 / 2020

ABSTRACT

Although the idea of using Neural Networks technology for Financial Time Series prediction is an old one, the abundance and availability of stock-price data, including high-frequency (intra-minute) data has given an additional impetus to this fledgling field of study. Nevertheless, the study and applications have focused on the hedge-funds and brokerage operations of investments banks and very little academic attention has been devoted to the day-trading activity aiming at the accumulation of short-term incremental gains – still considered as a retail activity, similar to betting. The purpose of this research is precisely to investigate the possibility to use the sound time series smoothing techniques of feedforward Neural Networks along with elementary but powerful Classification Neural Networks techniques to produce a decision-aid system for intraday trading decisions. The basic design of the study consisted in presenting a new typology of intraday patterns and apply it to the 126 trading days of the first half of 2020 for the SP500 index, using intraday, minute-by-minute prices. While applying this methodology generated questions for further research, the major finding of this study was that when a price trend was soundly established during the first half of a trading day session, more often than not (in 57 cases versus 31) the trend continued till the end of the trading day. This result, which translates into the possibility for the trader to engage in short-term profitable trades, is non-trivial, though needs more past data to be consolidated. Further research into the conditions prevailing in other markets, before a trading day begins, could also prove useful.

Contents

1. INTRODUCTION

Calibration of time-series models through filtering is a popular research area in econometrics and financial engineering [Hamilton, 1994], [Pollock, 1999], [Kim and Nelson, 1999], [West and Harrison, 1999], [Durbin and Koopman, 2001], [Harvey, 2001], [Shadbolt and Taylor, 2002], [Franses and Dijk, 2003], [Koop, 2006], [Harvey and Koopman, 2009]], [Shumway and Stoffer, 2011]. This research review reports preliminary investigations into processing financial time series using non-linear neural networks treated with Kalman filters, which are more powerful models than the popular econometric linear models.

The particular models studied here are based on Time-Delay Neural Networks (TDNN) [Haykin, 2009] trained by Extended Kalman Filtering [Anderson and Moore, 1979], [Haykin, 2001], [Durbin and Koopman, 2001], [Harvey, 2002] which makes them proper dynamical tools for time series modeling that explicitly account for the time parameter.

There are two major contributions in our research: 1) design of a Statistical TDNN (STDNN) architecture and implementation of a robust backpropagation algorithm for its training assuming the heavy-tailed Student-t distribution; and 2) formulation of a robust filtering algorithm for sequential estimation of studentized STDNN. First, we develop a Statistical Time-Delay Neural Network architecture (network topology) which extends the standard TDNN that infers the mean of the targets with another secondary network that infers the variance. This allows us to model not only the unknown data generating function but also simultaneously to quantify the uncertainty of the output for each network input. Second, we develop a filtering algorithm for robust sequential STDNN modeling of time series using the Student-t distribution. The objective is to achieve improved modeling of noisy (real-world) time series with sophisticated versions of non-linear neural networks. That is, instead of the widely used Gaussian model for nonlinear regression we use a more flexible noise model based on the Student-t-distribution. The parameters of the t-distribution such as the degrees of freedom are also estimated online using an incremental Expectation Maximization (EM) algorithm [McLachlan and Krishran, 1997],[Neal and Hinton, 1995]. The adaptation of the degrees of freedom of a t-distribution is of crucial importance because it helps to mitigate the effects of outliers, and it helps to avoid inappropriate weight changes due to measurement defects.

Previous research [Briegel andTresp, 2000], [Ting et al., 2007], [Särkkä, 2013] reported some experimental results using the t-distribution as a noise model and suggested that this may lead to stable online learning algorithms and can outperform state-of-the art online learning methods like the extended Kalman filter. Here we continue these ideas with further explorations into robust learning of non-linear temporal neural networks, however we adopt directly the Student-t distribution (that is, not its approximation) for learning all parameters and hyperparameters in the model. More precisely, Section 2 presents a derivation of a robust backpropagation algorithm for training extended STDNN, and Section 3 presents a robust filter for overcoming the effects of outliers which is elaborated for the STDNN. A distinguishing characteristic of the proposed approach is that all developed algorithms are fully incremental, that is they process the data online immediately after their arrival in time. Section 4 deals with modelling financial

time series and the taking into account financial data. Section 5, constitutes the experimental part of this study and proposes an intraday trading model based on Kalman Filter. Last, Section 6 hints to avenues of future research.

1.1. TIME SERIES MODELING

Time series have various applications in fields as diverse as clean energy production and finance [Mills, 1991, 2002], [West and Harison, 1999], [Pollock, 1999], [Kim and Nelson, 1999], [Shumway and Stoffer, 2000], [Chatfield, 2005], [Särkkä, 2013]. For example, time series depicting wind movement are used by the electrical energy practitioners to fine-tune forecasts of production.

This research focus on financial time series, which is an increasingly active area of research [Taylor, 1986], [Franses and van Dijk, 2003], [Durbin and Koopman, 2001], [Mills, 2002], [Tsay, 2005]. Specifically, we will try to analyze stock-market time series by taking advantage of the contribution of neural networks in the field [Deboeck et al., 1994], [Zapranis and Refenes, 1999], [Shadbolt and Taylor, 2002], [McNelis, 2005], [Nikolaev and Iba, 2006].

Time-series modelling may be viewed as an inductive learning problem [Nikolaev and Iba, 2006]. The learning task is to identify the regularities among a given series of measurements (points): …, x_t, x_{t+1}, x_{t+2}, …., sampled at discrete time intervals. The goal is to find out how future points depend on points from the past. In the ideal case the behavior of the data generator can be described by differential equations, but in practice knowledge of such mathematical modelling is not available. This is why, efforts to discover plausible descriptions of the unknown data source are made for the unknown parameters of a pre-selected model.

Here we use input vectors created from the given observations assuming embedding dimension d, and delay time τ [Takens, 1981], [Gershenfeed and Weigend, 1994]:

$$x = [\, x(t-\tau), x(t-2\tau), \ldots, x(t-(d-1)\tau)] \qquad (1)$$

that is lagged, sliding window vectors from $d\tau$ nearest previous points starting at point *t*. The delay time is a positive number $t>0$, here in all experiments *t* is used. The dependent variable is the immediate neighboring point to the start point $y= f(x)$.

The Takens' Theorem

A dynamical system can be described as a succession of states over increments of time. The contribution of a theorem due to Takens [Takens, 1981] is that given a sufficient number of observations we can reconstruct the dynamical system that has generated the series. Thus, one can extract useful information about the phases of a dynamical system and try to make predictions about future states [Weigend and Gershenfeld, 1994].

According to the Takens' theorem vectors of past time series data contain sufficient information to reconstruct the behavior of the unknown underlying system behavior. These vectors of delayed components are supposed to be degrees of freedom of the system that could have generated the time series. That is, the evolution of the system can be modelled by nonlinear functions of delay vectors. A key issue is the embedding dimension, that is, the number of the

components in the vectors that have to be selected in order for the system to converge toward its unknown yet original form.

While we know that each vector should include different information on the model, the separating delay time is very important: should it be too short then it would be polluted by noise; should it be too large, it becomes less informative, therefore less useful. Different algorithms exist for calculating the optimal global embedding dimension [Abarbanel, 1996], though the determination of the embedding dimension is still an active area of research.

1.2. NON-LINEAR VS. LINEAR MODELS

Linear time series models are straightforward to implement [Hamilton, 1994]. The penalty for this convenience is that they may be entirely inappropriate for even moderately complicated systems. Below we will present systems that are assumed to be linear and stationary.

Moving Average (MA) Models

The simplest model describing the relations between lagged exogenous inputs from an external series and the current observations is the Moving Average (MA) model. A model which is a combination of lagged inputs is called a Moving Average model. Formally, given vectors of a predetermined number of points from an external series $[u_{t-1}, u_{t-2}, \dots, u_{t-N}]$, the relation between these input points and the given observations y_t is defined by [Hamilton, 1994]:

$$y_t = \sum_{n=1}^{N} b_n u_{t-n} + b_0 = b_1 u_{t-1} + \cdots + b_N u_{t-N} + b_0 \quad (2)$$

where b_1, b_2,…, b_N are the parameters (weights) in the model, *N* is the selected dimension, and the dependent variable y_t is different from the inputs, that is we have $y_t \neq u_t$.

This is a linear MA model in the parameters *b* which we are going to call here weights in connectionist neural network parlance. In a general neural network model this relationship can be nonlinear [Haykin, 2011].

The linear MA models are popular in econometrics and time series analysis, where they are also viewed as smoothing filters [Durbin and Koopman, 2001]. Such filters are also known as finite impulse response (FIR) filters [Weigend and Gershenfeld, 1994], because when the input becomes zero after *d*-steps the output becomes also zero. The FIR filter is a memoryless model.

Autoregressive (AR) Models

Models that describe the relationship between previous (lagged) points from a series and the next series point are called Autoregressive (AR) models. Let $[y_{t-1}, y_{t-2}, \dots, y_{t-M}]$ be the vector from a selected number of points from the given series (for example using the Takens'

theorem), and y_t be the next point which serves as dependent variable. The linear AR model describing the mapping between the dependent variable and the input is [Hamilton, 1994]:

$$y_t = \sum_{m=1}^{M} w_m y_{t-m} + w_0 + \varepsilon_t \quad (3)$$

where w_1, w_2,…, w_M are the weight parameters weights in the model, and ε_t is the noise term. Usually the noise is assumed to be zero-mean independent Gaussian.

This AR model is also known by engineers as infinite impulse response (IIR) [Weigend and Gershenfeld, 1994] model because it continues to produce output even after the input terminates, due to the effects from the noise ε_t which is crucial for the life of an AR model.

Autoregressive Moving Average (*ARMA*) *Models*

The most general time series models combine autoregressive and moving average parts together into ARMA models. An ARMA(*M*,*N*) model is defined as follows [Hamilton, 1994]:

$$y_t = \sum_{m=1}^{M} w_m y_{t-m} + \sum_{n=1}^{N} b_n u_{t-n} \quad (4)$$

where m and n are respectively the orders of the submodels. That is m is the order of the AR part and n is the order of the MA part.

The ARMA models are linear models which have found practical applications in various fields, including time series analysis, econometrics and empirical finance [West and Harrison, 1999], [Harvey and Koopman, 2009]. The inference of such models involves finding the order of the model (that is the lagged dimension of each of the autoregressive and moving average parts) and, next, estimation of the parameters. When searching for the model order one should be careful not to choose an order larger than the dynamics of the underlying system, otherwise the model will not forecast well. When learning the model weights one should avoid overfitting, otherwise the inferred model will not generate useful predictions.

A problem arising in time series modelling is the choice between a linear or non-linear model for the situation we want to represent [Franses and van Dijk, 2003], [McNelis, 2005], [Tsay, 2005]. Although the linear models feature clear, thoroughly studied properties and they are easy to understand and implement, many practical data may not admit plausible representation by them. Tests for nonlinearity should be applied to detect the inherent data characteristics, and if inherent nonlinearities are found in the data nonlinear models should be selected.

Nonlinear Models

Stock-market time series are typically described by mathematical functions dependent on past values. That is, prices at *t* will exert an influence on price at *t+1*. To capture such phenomena we try to regress values of *t* on *t-n*.

An important learning problem when dealing with time series is how to choose between linear and non-linear models for a task [Box and Jenkins, 1970]. Linear models are very popular in econometrics and finance [Hamilton, 1994]. On the other hand, the theory of dynamical systems indicates that even irregular series produced by deterministic dynamics often can be

reconstructed easily with the less popular nonlinear models. Various tests allow us to decide if a linear / non-linear model is required for a specific modeling task [Nikolaev and Iba, 2006].

The popularity of linear models is due to a great degree to the sound theory behind them, but they still need preprocessing to handle trends and seasonalities for example. Nonlinear models are especially suitable for describing series with oscillations and stochastic effects.

The particular models studied here are Time-Delay Neural Networks (TDNN) [Clouse, Giles Horne and Cottrell, 1997]. The name TDNN is selected in this research to denote a general class of autoregressive nonlinear time series models, although various other notions have been also used in similar research, namely: NNFIR [Wan, 1993], NARX [Lin, Horne, Tino, and Giles, 1996], TLFN [Haykin, 2009] and NARNN [Nørgaard et al., 2000]. Common in all these connectionist architectures is that they serve as temporal models, that is they process temporal information. Temporal in this case means that the inputs are ordered in time. They are especially adequate for describing time series, or, in other words, to capture time dependencies using a tapped delay inputs. The tapped delay input line is considered a kind of memory that helps for the proper temporal learning. The other kind of dynamic neural networks, namely the recurrent networks are not considered in this study due to their well-known training difficulty arising because of the vanishing with the time temporal gradient [Lin, Horne, Tino, and Giles, 1996].

It should be noted that our particular neural network is actually a subclass of TDNN corresponds to the so-called Input Delay Neural Network (IDNN) [Clouse, Giles Horne and Cottrell, 1997]. Their distinguishing characteristic is that time-delayed signals from the inputs are only passed to the hidden nodes, while the hidden node outputs are passed to the output in ordinary manner. The most recent data passed to our TDNN play the role of a tapped delay line, which facilitates the leaning of temporal relationships between the inputs. Another interesting feature of our TDNN is that it can be easily treated as a dynamic state-space model, and trained with proper sequential estimation algorithms. This is what makes them efficient for financial time series processing. All these characteristics motivate our research into TDNN.

Financial Applications of Linear and Nonlinear Models

The linear models trained by Kalman filtering are very popular nowadays in economics and finance [Hamilton, 1994], [Harvey, 2001], [Harvey and Koopman, 2009]. Such models have been considered not only for traditional economic tasks, such as modeling inflation, unemployment, risk management interest rate modeling [Kim and Nelson, 1999], consumption [Durbin and Koopman, 2001], and portfolio selection [Nikolaev and Iba, 2006] but also recently for the design of algorithmic trading strategies, pairs trading [Elliott et al., 2005] and statistical arbitrage [Burgess, 1999], [Avellaneda, 2010] as well. These applications are mainly due to the powerful representation potential of dynamic state space models, which can capture short-term and long term movement characteristics, and also due to the flexibility of their training filters which allow to accommodate efficiently the next data item arriving sequentially in time. .

Some well known economic applications of dynamic models include:

-interest rates modeling [Kim and Nelson, 1999]
-modeling inflation [Harvey and Koopman, 2009]
-modeling unemployment [Harvey and Koopman, 2009]
-risk management and volatility estimation [Kim and Nelson, 1999], [Danielsson, 2011]
-optimal asset allocation [Pitt and Shephard, 1999], [Aguilar and West, 2000]

-modeling industrial sales [West and Harrison, 1999]
-tactical asset allocation [Zapranis and Refenes, 1999]
-yield curve arbitrage [Refenes et al., 1997]

Some well known applications of dynamic models for making trading algorithms include:

-pairs trading [Elliott et al., 2005]
-statistical arbitrage [Burgess, 1999], [Avellaneda, 2010], [Montana et. al. , 2009]
-option pricing [deFreitas et al., 2000], [Tino et al., 2001]

1.3 PROCESSING FINANCIAL DATA

Our intention to conduct research into modeling financial time series respects the well known Random Walk Hypothesis (RWH) [Malkiel, 1973], and Efficient market hypothesis [Fama, 1970], and goes further trying to describe trends in market movements without contradicting these theories. The RWH states that stock market prices move in unpredictable way, and we agree with that, that is we assume that prices have random behavior, ie the price changes/increments are independent, but at the same time we believe that trends in short term changes can be seen and they can be described functionally to a great degree. This directs our attention to the use of concepts from the technical analysis for identification of price movement direction, momentum and acceleration through various kinds of oscillators for example [Kaufman, 2013]. We are modeling not the prices directly, but rather the changes in their movements described by oscillators computed from given data.

The Efficient Market Hypothesis (EMH) [Fama, 1970] states that markets are rational and accommodate fully any new information, so even if predictions are used for speculation they will also be absorbed by the market and the price behavior will return back and will follow random walk again. We agree to a great extent with the EMH as well, but we also believe that short-term trends could be detected and could be exploited for the generation of trading signals for scalping (that is for gaining profits from short-term price changes if captured properly). That is why, our objective behind modeling the trends in oscillators from prices is primarily the generation of useful trading signals that provide information about the direction of price movements only in the near future.

Technical Analysis

Here we adopt concepts from the technical analysis [Murphy, 1999], [Brown, 2012], [Kaufman, 2013], [Pring, 2014] for extraction of trend patterns from financial price series. Of course we are not going to use archetypal price patterns, but we will use proper technical indicators. The technical indicators are price transformations that typically convert the nonstationary price movement into tractable stationary signal which we believe can be modeled to a certain degree of accuracy by contemporary non-linear neural network models. We intend to use different technical indicators, like the RSI, the Williams% *K*, the DPO etc., in order to find out which of them seem more amenable to learning and prediction using the STDNN network model.

Processing Returns on Prices

Most of the research in neural networks for financial modeling [Deboeck et al., 1994], [Zapranis and Refenes, 1999], [Shadbolt and Taylor, 2002], [McNelis, 2005], [Nikolaev and Iba, 2006] attempt to learn a nonlinear model of the returns on prices, so as to avoid direct processing of nonstationary price series they preprocess them and convert into stationary.

The log-returns from a series of prices x_t $(1 \leq t \leq T)$ are computed as follows:

$$l_t = \log x_t - \log x_{t-1} \qquad (5)$$

where the assumption is that $x_t = x_{t-1} + \epsilon_t$, $\epsilon_t \sim N(\emptyset, \sigma^2)$ is white noise.

It should be noted that various research efforts have been made to send as inputs to the neural networks various variables generated by technical indicator transformation techniques, but as given data there were still considered directly the returns from prices (these are commonly

called neural network hybrids). We think that the returns on prices are unpredictable, and our intention is to use as target output data produced by oscillators, like the RSI, the Williams%*K*, and the DPO. That is why, we incorporate a special detrending layer in out proposed network.

A challenging problem remaining after data pre-processing are the remaining outliers. Large amount of financial econometric research has been spent on handling outliers on financial time series. It should be emphasized that our proposal for robust STDNN network training using robust sequential estimation (or also called robust filtering) is applicable to any financial series.

Feeding detrended data as inputs to the STDNN can be envisioned as passing cleaned from noise to certain extent price series, and simultaneously with that white enhanced trend characteristics, such as relative trend strength, rate of change etc. Looking from the previous angle these technical analysis transformations are also a kind of pre-processing techniques (leading usually to stationary oscillating series).

When developing a neural network model for any financial series there are several design issues that have to be addressed carefully: 1) how to select the input variables; 2) how to design the neural network architecture, including the selection of the number of hidden nodes; 3) which training filter to use, that is to find the particular training algorithm; 4) how to perform post-processing after back testing, this involves eventual network pruning for overfitting avoidance. The evaluation of the network performance is made here with various well known statistical and econometric measures [Miazhynskaia et al., 2005], such as: NMSE, NMAE, HITs, etc.

1.4. AIMS AND OBJECTIVES

The main objective of this research is to investigate the application of novel STDNN models and training algorithms for NN which facilitate the generation of useful trading signals from financial price series. While most of the current work dedicated to generation of trading signals exploits the possibility to predict returns with neural networks [Shadbolt and Taylor, 2002], [McNelis, 2005], [Nikolaev and Iba, 2006], which forecasts produce the trading decisions, our work is innovative in the sense that we attempt to forecast the evolution of technical indicators (calculated over prices) such as Moving Averages, to take just the simpler ones (but still very powerful).

While the theoretical part of this work focuses on Neural Networks smoothing of time series, this serves a purpose: smoothed intraday time series are easier to fit into patterns that are recognized by trader – might he be a human operator or a machine.

The study culminates with the proposal of a typology of intraday trading patterns (s. Empirical Section) and their application against the 126 trading days of the first half 2020.

It is important to underscore the synergetic role of the two parts of this study: the theoretical part reviews the latest NN technologies and provides a scientific background to the study of trading patterns which otherwise would have been reduced to the rank of archetypal figures of the so-called “technical” or “chartist” analysis that, while widely practiced, never managed to establish itself as an academic field precisely because of the lack of underlying theory.

To summarize, this study pursues three specific objectives:

-(A) Provide an extensive review of existing NN technologies and their use for financial series smoothing (Sections 1 to 4)

-(B) Propose an original typology of intraday trading patterns (Section 5)

-(C) Test the possibility to apply (B) to a particular pattern of intraday time series, the Ascending / Descending Trend and examine if useful trading signals can be generated.

2. STATISTICAL TIME-DELAY NEURAL NETWORKS

Neural Networks (NN) are non-linear models [Hornik, 1989] which are suitable for time series modeling [Weigend and Gershenfeld, 1994], [Pham and Liu, 1995], [Haykin, 1999], [Zapranis and Refenes, 1999], [Nørgaard et al., 2000], [Frances and Van Dijk, 2003], [McNelis, 2005], [Nikolaev and Iba, 2006], [Haykin, 2009]. A neural network model is a universal approximator as it can approximate any bounded continuous function up to a desired accuracy when enough training data and hidden neurons are available [Hornik, 1989].

One problem however of traditional feed-forward neural networks for time series processing is that the network model is static, i.e. there is no time dimension. The simplest way to render a static neural network model suitable for temporal processing is to feed the time directly into the network along with the inputs. That is why, here we accommodate the time into feed forward architecture through a delay space embedding, also known as a sliding window or tapped delay line. The sliding window widens the input space and orders information from the present to a pre-specified distance back in time $x_t = [x_{t-1}, x_{t-2}, x_{t-3}, \dots]$ [Weigend et al., 1990].

Another problem of traditional feed-forward neural networks is that their specification typically assumes normal, Gaussian noise distributions. The normal noise density however can not approximate well practical time series data featuring higher skewness and excess kurtosis. In order to achieve a better fit to the data one has to adopt heavy-tailed noise densities, like the Student-*t* density for example.

Our research proposes a studentized statistical time-delay neural network (STDNN) which learns time-dependent functions of each the mean and the variance of the data distribution. The developments here include: 1) design of a STDNN network architecture (topology) with two interdependent parts having separate output nodes: one that infers the mean and another that infers the variance of the target distribution; 2) derivation of gradient vectors for updating all weights in both parts of the topology using a special cost function for a heavy-tailed noise model using directly the Student-*t* pdf; and 3) formulation of a robust backpropagation algorithm for incremental (online) gradient-based training of both parts of STDNN. Thus, we arrive at an algorithm that can learn not only from homoscedastic but also heteroskedastic time series.

Motivation for Developing Temporal Neural Networks

The traditional Multilayer Perceptron (MLP) network [Werbos, 1974], [Rumelhart et al., 1986] and its temporal version Time-Delay Neural Network (TDNN) studied here, are flexible nonlinear models that are suitable for time series modeling due to several reasons:

- they are universal approximators, which means that if the data can be represented by a mathematical function it can be found by fitting such a network model;
- they are reliable to train, which means that their training will always converge to a weight parameter vector, even if there are mild outliers and discrepancies in the data;

The MLP networks can be classified into two sorts based on the directional flow of information within the network, namely “feedforward” and “recurrent” neural network models. In a feedforward network, a processing element can send information only to units which it does

not receive information from directly or indirectly resulting in a static mapping from its input to its output space. Recurrent networks are suited for temporal tasks due to their intrinsic memory resulting from the configuration of the network whereas feedforward networks are not specifically designed for temporal processing because of their lack of internal memory. The memory plays a key role in temporal sequence processing and thus architectural considerations specifying network topology and the activation dynamics of the processing elements is essential when dealing with temporal patterns. Therefore, we can either add memory to a feed-forward network, or use recurrent neural networks (RNN) with inherent memory. Previous research in our lab [Mirikitani, 2010] has found that not only RNN networks are difficult to train, but also they lead to unstable learning performance and not excellent results on practical data.

We study here feed-forward TDNN networks, which implement non-linear functions given time-series data. First, the TDNN network is a nonlinear autoregressive model which can describe a large class of deterministic systems using finite length input windows with its hidden weights that serve as the hidden states of the series dynamics. Second, the TDNN is a dynamic states space model which is particularly suitable for time series modeling because it implements a generative function of weights that capture various descriptive aspects of the series in addition to the autoregressive information implicitly passed by the given input variables.

All these properties provide us an unambiguous motivation to use TDNN for financial time series modelling. Essentially they suggest that if a model, of the unknown function that have generated the given time series data, this model is learnable with TDNN. Further, we extend the TDNN with an additional preprocessing layer of units for data detrending. Thus, possibly non-stationary series are transformed into stationary series to enable their proper processing.

2.1. STATISTICAL TEMPORAL NETWORKS

2.1.1 Extended Temporal Networks

This research develops a Statistical Time-Delay Neural Network (STDNN) that learns both the mean and the variance of the data distribution. Following a similar research into static multilayer networks [Nix and Weigend, 1995], [Nikolaev and Iba, 2006] a secondary network is added, after the preprocessing layer, to model the data variance. Thus, we have a primary network that infers the mean and a secondary network which infers the variance. Both these networks have MLP topologies.

The STDNN network is an adaptive nonlinear model whose components are:
-inputs passed to the network;
-processing units called neurons, each having an activation function (usually this is the sigmoidal function) which have derivatives that are computed fast;
-connections between the neurons, each associated with a corresponding weight.

The ingredient MLP networks are feedforward in the sense that the signals are send forward through the networks and there are no feedback signals from other neurons. Since this actually

leads to static mappings, to make them temporal, time lags are added in order to allow for the tracking of time dependencies. That is, when we pass time-delayed inputs we add to the overall model a temporal dimension, and the flow of information in the architecture is still only in forward direction. In other words, for processing temporal data the network is given a (temporal) memory structure that takes into account past values from the time series. In each layer of a feedforward network there is a row of neurons and between the layers there is a connection represented by an adaptable weight. In our case we are concerned with auto-regressive process, so there is a single output unit that produces the network output.

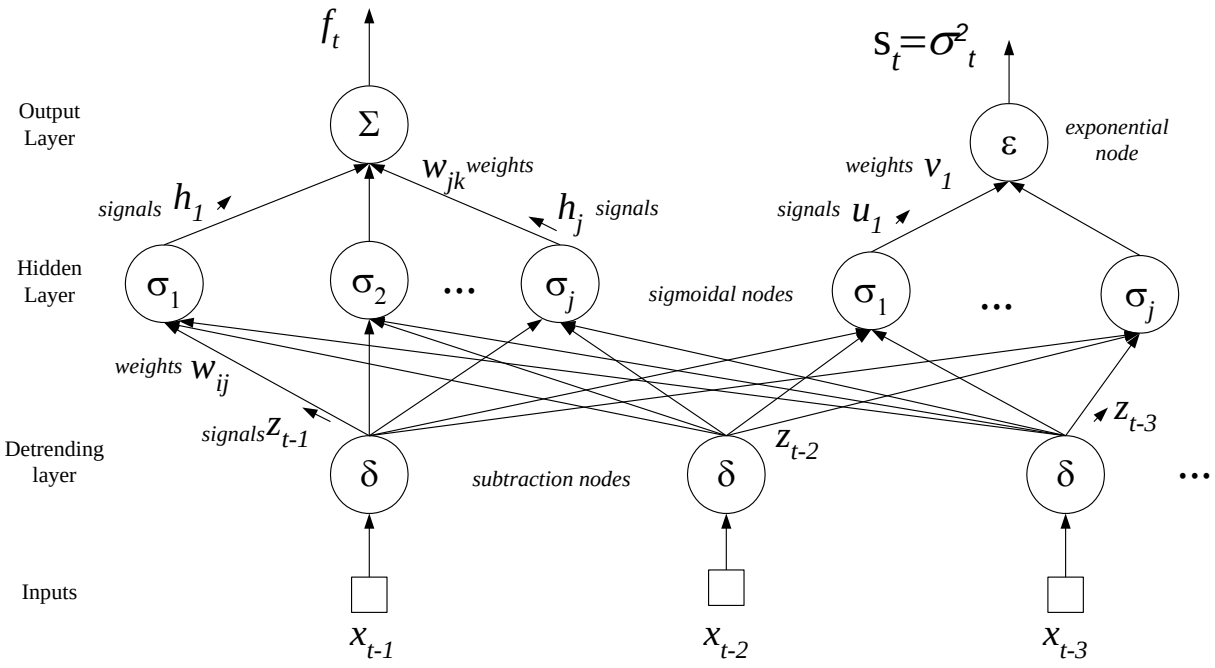


Figure 1. Connectionist architecture of the feedforward STDNN network model.

2.1.2 The Feedforward STDNN Architecture

The STDNN produces a time-varying estimate of the error distribution around its mean. While the primary TDNN output yields the mean: $f_t = E[y_t|x_t]$), the secondary output node yields the noise variance: $s_t = \sigma_t^2$ assuming that it is not constant but dependent on the inputs. Thus, the conditional probability density of the output errors is inferred as a function of the input data. The STDNN architecture is given in Figure 1.

The neurons in feed-forward networks are organized into layers, namely the input, hidden and output. The processing elements are usually fully connected to the units in the next layer. In feedforward networks the information flows in one direction from the input layer to the output layer. In each layer there is a row of neurons and between the layers, there are connections each associated with an adaptable weight.

A neuron combines all of the input signals coming into the unit along with a bias value into a weighted sum and performs a weighted summation of the inputs. The neuron output is a function of the weighted sum passed through an activation function. The output is then passed to other units in the next layers, or to the output of the network.

Let the given data be: $\{x_t, y_t\}(t = 1..T)$, where x_t is the input lagged vector:

$$x_t = [x_{t-1}, x_{t-2}, x_{t-3}, \dots] \qquad (6)$$

and y_t is the given target (that is the next point in the series). This is a sliding window and its delay embedding is of crucial importance: it orders the information at each step by including new data and discarding old data.

We elaborate a studentized version of the STDNN assuming that the noise comes from the heavy-tailed Student-*t* distribution, that is we assume that the residuals are zero-mean with studentized probability density $r_t \sim St(0, \sigma_t^2, n)$ having variance (precision) σ_t^2 and degrees of freedom *n*. This studentized STDNN is actually a kind of density neural network that learns simultaneously two models of the mean and the variance of the data using the Student-*t* probability density function.

Let the weights in the mean part of the network $\{w_{ij}, w_j\}$ $(i = 1 \dots d, j = 1 \dots H)$, as well as these in the variance network $\{v_{ij}, v_j\}$ $(i = 1 \dots d, j = 1 \dots H)$, be arranged into vectors:

$$w_t = [w_1, w_2, w_3, \dots] \qquad (7)$$

$$v_t = [v_1, v_2, v_3, \dots] \qquad (8)$$

The formal model of the *studentized* STDNN neural network that we develop is:

$$y_t = f(x_t, w_t) + r_t, \qquad r_t \sim St(0, \sigma_t^2, n) \qquad (9)$$

$$f(x_t, w_t) = \sum_{j=1}^{H} w_j \, \sigma\left(\sum_{i=1}^{d} w_{ij} \delta(x_{t-i}, p)\right) \qquad (10)$$

$$s_t^2 = \varepsilon\left(\sum_{i=1}^{d} v_j \, s\left(\sum_{i=1}^{d} v_{ij} z_{t-i}\right)\right) \qquad (11)$$

where the second equation defines the model of *mean*, the third equation defines the model of the *variance*, σ denotes the sigmoid function, ε denotes the exponential function, and δ is a detrending function with pre-defined period *p*. In order to simplify the notation sometimes we will use s_t instead of the instantaneous variance s_t^2, and also $f(x_t, w_t)$ instead of $f_t(x_t, w_t)$. The sigmoid activation function $\sigma(k)$ is: $\sigma(k) = 1/(1 + e^{-k})$.

Detrending Layer

A distinguishing feature of the proposed STDNN is that it uses a special common layer that performs detrending of the data. Here we consider the following formulae:

$$z_t = \delta(x_{t-i}, p) = x_{t-i} - MA(x_{t-i}, p) \qquad (12)$$

$$y_t = \delta(x_t, p) = x_t - MA(x_t, p) \qquad (13)$$

where *MA* denotes moving average with period *p*: $MA(x_t, p) = \sum_{i=1}^{p} x_{t-i}$. Both equations use the same period for averaging.

Note that here both the inputs as well as the target values are detrended, this means that we actually model the detrended series not the given series directly. An advantage of this approach is that it enables us to handle non-stationary series, like these from financial prices for example.

Primary Mean-TDNN Architecture

The *j*-th hidden neuron in the mean-network operates as follows:

$$k_j = \sum_{i=1}^{d} w_{ij} z_{t-i} \qquad (14)$$

where w_{ij} are the weights on connections toward the *j*-th neuron in the hidden layer, z_{t-i} are the detrended inputs, and the input index *i* ranges up to the selected embedding dimension *d*.

The *j*-th hidden unit output activation is computed through the activation function which is typically a sigmoid σ, so we have:

$$h_j = \sigma(k_j) = \sigma\left(\sum_{i=1}^{d} w_{ij} z_{t-i}\right) \qquad (15)$$

The output of the network f_t is computed by weighted summation over the hidden node outputs provided by the neurons in the hidden layer:

$$f_t = \sum_{j=1}^{H} w_j h_j \qquad (16)$$

For auto-regressive problems typically a single output unit produces the network output f_t.

Secondary Variance-TDNN Architecture

The output node of the secondary variance-network also accepts signals from the inputs through the common detrending layer. The extension features full connectivity, that is, every detrended input is passed along a corresponding link to every node in the additional hidden layer, whose outputs are next passed to the second output node. The expanded topology has the same sigmoidal activation functions as these in the original mean-part.

The additional hidden layer uses sigmoidal activation functions to filter out the weighted summations of the incoming detrended signals:

$$u_j = \sigma\left(\sum_{i=1}^{d} v_{ij} z_{t-i}\right) \qquad (17)$$

where z_{t-i} are the signals from the detrending nodes, v_{ji} are the weights on the connections to the extended hidden nodes.

However, the secondary output node transforms the weighted summation by the exponential function in order to guarantee production of positive values only:

$$s_t^2 = \varepsilon\left(\sum_{j=1}^{H} v_j u_j\right) \qquad (18)$$

where v_j are the weights on connections feeding the second output node, u_j are the outputs of the *H* additional hidden nodes, and ε denotes the exponential function.

The secondary network is also an MLP with universal approximation capacity. Overall it has a similar structure from the primary network, and has enough power to produce a good estimate of the variance. The secondary network learns the model uncertainty of the primary network as it reflects its performance by accepting its inputs through the detrending nodes.

2.2. TRAINING STUDENTIZED STDNN NETWORKS

2.2.1 Gradient-descent Network Training

Neural network training is a search problem. We are searching for the set of weights that will cause the network model to achieve the lowest global error for a training set. If we have an infinite amount of computation resources, we will try every possible combination of weights and search for the one leading to the best global error. However, we do not have unlimited computing resource, so we have to use some sort of shortcut. Each training method is a way of finding an optimal set of weights without doing exhaustive search, which is computationally impossible. That is why, we conduct guided search. The search in the space of weights is guided by the error gradients with respect to the weights.

The gradient is the slope of the error function at a specified weight. The gradient is given by the derivative of the cost function curve at that point. This curve tells us something about how steep the error surface at the given weight. Used within a training technique, it provides insights into how the weight should be adjusted for attaining a lower error. We compute the derivatives of the cost function with respect to the weights in the network through a technique called backpropagation [Werbos, 1974], [Rumelhart et al., 1986], [Werbos, 1994], [Haykin, 2009].

Studentized Training Criterion

Our research proposes robust training of STDNN with a studentized cost function, that is assuming the Student-*t* probability distribution function as a training objective to derive weight training equations for all the parameters in both parts of the network. This is important as in practice real-world data are often corrupted by observational noise with non-normal characteristics. Adopting a Student-*t* cost function allows us to handle data that come possibly from heavy-tail distributions which have longer and thicker tails. Training the double, extended STDNN network architecture means that actually we are going to infer online the time-varying mean and time-varying variance of the data.

Our development of a studentized training algorithm for error mean and variance learning, relies on two assumptions [Nix and Weigend, 1995]: 1) that the noise in the data obeys the Student-*t* distribution, and 2) that the errors are statistically independent.

The coherent simultaneous training process of both the first and the second network parts aims at maximization of the likelihood of all data points:

$$C_T = \prod_{t=1}^{T} p_s(y_t|Y_{t-1}) \qquad (19)$$

where $p_s(y_t|Y_{t-1})$ is the conditional Student-*t* probability density function of the data, and Y_{t-1} denotes all data arrived up to moment *t-1*.

We consider the following Student-*t* distribution formula:

$$p_s(y_t|Y_{t-1}) = \frac{\Gamma((n+1)/2)}{\sqrt{\pi n s_t}\Gamma(n/2)}\left(1+\frac{e_t^2}{ns_t^2}\right)^{-\frac{(n+1)}{2}} \qquad (20)$$

where s_t^2 is the precision, the model error is $e_t = y_t - f_t$, the parameter n $(n > 2)$ is the degree of freedom of the Student-*t* distribution, and $\Gamma(\cdot)$ denotes the Gamma function.

When the degree of freedom parameter n approaches infinity the model approaches the Gaussian noise model, so this parameter n may serve as a knob for tuning the degree to which the model accounts for the regularities in the observations. This parameter *n* can either be pre-defined for simplicity, or found by optimization as explained in the next chapter. When the parameter is fixed in advance we begin with *n=100* to achieve the same fit as with the normal distribution (just to calibrate the performance of the training algorithm), and next in the experiments we use *n=4-10* for practical work with heavy-tailed noise.

The training objective is maximization of the cost function C_T, which is usually performed by minimization of its negative log-likelihood. Taking the negative log-likelihood of the cost function we arrive at the following instantaneous criterion suitable for online training:

$$\begin{aligned} E_t &= -\log C_t(y_t|Y_{t-1}) = -\log p_s(y_t|Y_{t-1}) \qquad (21)\\ &= -\log\Gamma\left(\frac{n+1}{2}\right) + \frac{1}{2}\log s_t^2 + \frac{1}{2}\log(n) + \log\Gamma\left(\frac{n}{2}\right) + \frac{n+1}{2}\log\left(1+\frac{e_t^2}{ns_t^2}\right) \end{aligned}$$

where the constants are omitted.

The weight update rules for backpropagation training of the extended STDNN are obtained by seeking the minimum of this negative log-likelihood criterion, that is, by differentiating this negative log-likelihood criterion with respect to the weights, equating the resulted expression to zero, and solving it for the free variables. This procedure is used for both the first STDNN output generating the mean, as well as for the second network output generating the variance as they are mutually interdependent.

Calculating the Gradients of the Cost Function

We will now explain how to calculate the gradient vector of the error derivatives with respect to all weights in the neural network. It should be noted that here we obtain the analytical temporal derivatives of the cost function with respect to the weights that take into account the time, and are especially suitable for incremental (online) training. Although we next plug these derivatives into the backpropagation training algorithm which iterates many times over the same data, this possibility for incremental learning facilitates not only proper handling of time series data, but also helps for doing accurate forecasting.

The training procedure involves the calculation of the partial derivative for the output error at each output neuron with respect to its incoming weights. This can be accomplished using a method based on the chain rule from calculus.

The individual gradients are obtained by calculating partial derivatives of the instantaneous cost with respect to each individual weight using the chain rule as follows:

$$\frac{\partial E_t}{\partial w_t} = \frac{\partial E_t}{\partial f_t}\frac{\partial f_t}{\partial w_t} = \frac{\partial E_t}{\partial e_t}\frac{\partial e_t}{\partial f_t}\frac{\partial f_t}{\partial w_t} \qquad (22)$$

$$\frac{\partial E_t}{\partial v_t} = \frac{\partial E_t}{\partial s_t^2}\frac{\partial s_t^2}{\partial v_t} = \frac{\partial E_t}{\partial e_t}\frac{\partial e_t}{\partial s_t^2}\frac{\partial s_t^2}{\partial w_t} \qquad (23)$$

where the first equation is for the weights $w_t = [w_1, w_2, w_3, \dots]$ in the network that produces the mean $f_t = E[y_t|x_t]$, and the second is for the weights $v_t = [v_1, v_2, v_3, \dots]$ for the part of the network that produces the variance s_t^2.

2.2.2 Training the Mean Network Part

Training the mean network requires us to find the derivative of the instantaneous cost with respect to the output of the mean network part $\partial E_t/\partial f_t$. This derivative is obtained in two steps by taking the derivative of the instantaneous cost with respect to the residual error $\partial E_t/\partial e_t$ and next by taking the derivative of the residual error with respect to the output $\partial e_t/\partial f_t$. These two steps can be summarized as follows:

$$\begin{aligned}\frac{\partial E_t}{\partial f_t} &= \frac{\partial\left[(n+1)\log\left(1+e_t^2/(ns_t)\right)/2\right]}{\partial f_t} \\ &= \left(\frac{n+1}{2}\right)\left(\frac{1}{1+e_t^2/(ns_t)}\right)\left(\frac{2e_t}{ns_t}\right)(-1) \\ &= -\left(\frac{n+1}{2}\right)\frac{2e_t(ns_t)}{ns_t+e_t^2}\frac{1}{ns_t} \\ &= -\frac{(n+1)e_t}{ns_t+e_t^2} \qquad (24)\end{aligned}$$

The output gradient vector $g_t = \partial f_t(x_t, w_t)/\partial w_t$ contains the partial derivatives of the output $f_t(x_t, w_t)$ with respect to each individual weight from the vector $w_t = [w_1, w_2, w_3, \dots]$ in this network part defined as follows:

$$g_t = \frac{\partial f_t(x_t, w_t)}{\partial w_t} = \left[\frac{\partial f_t(x_t, w_t)}{\partial w_1}, \frac{\partial f_t(x_t, w_t)}{\partial w_2} \dots\right] \qquad (25)$$

where each component is derived using the chain rule:

$$\frac{\partial f_t(x_t, w_t)}{\partial w_t} = \frac{\partial f_t}{\partial h_t}\frac{\partial h_t}{\partial w_t} \qquad (26)$$

The output derivative for the hidden-to-output connections in the mean network is:

$$\frac{\partial f_t}{\partial w_j} = h_j \qquad (27)$$

Then, the delta rule for training the hidden-to-output weights w_j in the mean network becomes:

$$\Delta w_j = \eta \frac{\partial E_t}{\partial f_t} h_j \qquad (28)$$

where w_j are the hidden-to-output weights.
The derivatives of the input-to-hidden connections in the mean network are as follows:

$$\frac{\partial f_t}{\partial h_j} = w_j \qquad (29)$$

$$\frac{\partial h_j}{\partial w_{ij}} = h_j(1 - h_j) z_{t-i} \qquad (30)$$

Then, the delta rule for training the input-to-hidden connections in the mean network is:

$$\Delta w_{ij} = \eta \frac{\partial E_t}{\partial f_t} w_j h_j (1 - h_j) z_{t-i} \qquad (31)$$

where w_{ij} are the input-to-hidden weights.

This formulation of the backpropagation algorithm for training the STDNN network do not handle directly the given inputs, but rather their detrended versions as we are interested in learning from time series that can be highly nonstationary, like financial prices for example.

2.2.3 Training the Variance Network Part

Let the second output network node that models the variance s_t^2 has incoming connections from the hidden nodes weighted by v_j indexed by *j*, and the input-to-hidden weights be weighted by v_{ji} as given in Figure 1. The derivative of the considered instantaneous Student-*t* log-likelihood with respect to the variance s_t^2 is obtained as follows:

$$\begin{aligned}
\frac{\partial E_t}{\partial s_t} &= \frac{1}{2} \frac{\partial \log s_t}{\partial s_t} + \frac{\partial \left[(n+1) \log \frac{(1 + e_t^2/(n s_t^2))}{2} \right]}{\partial s_t^2} \\
&= \frac{1}{2 s_t} + \left[\left(\frac{n+1}{2} \right) \left(\frac{1}{1 + e_t^2/(n s_t^2)} \right) \left(\frac{e_t^2}{n} \right) \left(\frac{-1}{s_t^2} \right) \right] \\
&= \frac{1}{2 s_t^2} - \frac{1}{2 s_t^2} \frac{(n+1)}{(1 + e_t^2/(n s_t^2))} \frac{e_t^2}{n} \\
&= \frac{1}{2 s_t} - \frac{1}{2 s_t^2} \frac{(n+1) s_t^2}{(n + e_t^2/s_t)} \\
&= \frac{1}{2 s_t^2} \left[s_t^2 - \frac{(n+1) e_t^2}{(n + e_t^2/s_t^2)} \right] \qquad (32)
\end{aligned}$$

Since the derivative of the exponential function is its value, the output derivative for the hidden-to-output connections in the variance network is $ds_t/dv_j = s_t u_j$.

Then, the delta rule for training the hidden-to-output weights v_j in the variance network is:

$$\Delta v_j = \eta \frac{\partial E_t}{\partial s_t} s_t^2 u_j \qquad (33)$$

where η is the learning rate, and u_j is the signal on its connection from the additional hidden nodes. It has to be noted here that this rule uses the squared error e_t^2 produced at the output of the first node when the original STDNN is estimated with the same input.

The derivatives of the input-to-hidden connections in the variance network are as follows:

$$\frac{\partial s_t^2}{\partial u_j} = s_t^2 v_j \qquad (34)$$

$$\frac{\partial u_j}{\partial v_{ij}} = u_j(1 - u_j) z_{t-i} \qquad (35)$$

Then, the weights on the input-to-hidden connections in the variance network are updated by:

$$\Delta v_{ij} = \eta \frac{\partial E_t}{\partial s_t} s_t^2 v_j u_j (1 - u_j) z_{t-i} \qquad (36)$$

where v_{ij} are the input-to-hidden weights.

2.2.4 Phases of the Training Process

The extended STDNN network is trained with a version of the backpropagation algorithm which conducts gradient descent search in the weight space with the above learning equations. The second volatility network however is a nonlinear MLP which suffers from the problem of entrapment into suboptimal solutions. That is, the neural network training process may lead to weights which cause an error that is not optimal, rather it is an acceptably good local minima. In addition to this, the learning rules for the hidden to output node connections in both parts of the STDNN network are mutually dependent on the detrending layer. Because of these reasons, in order to avoid entrapment at suboptimal local optima on the error surface, the weight training is subdivided into three phases.

In the first phase, the mean STDNN network is trained to minimize the cost function according to the backpropagation technique, while the extended secondary variance part of the network is not trained, rather its weight parameters are kept fixed. This stage should be performed with only a subset of the training data so as to avoid eventual overfitting. The second phase uses a different subset of the given data to train the extended variance part of the network, also aiming at minimization of the cost function. This is implemented using the backpropagation algorithm with the learning rule for the second output, and the corresponding learning rules. In the second phase the weights in the mean part of the network remain unchanged. In the third phase the training objective is to minimize the common log-likelihood criterion by training simultaneously both parts of the STDNN network. During this third phase the weights in the mean part and the weights in the variance part are tuned each with their particular learning rules. The training proceeds until reaching the minimum of the log likelihood criterion; that is, until attaining a satisfactory low error.

Robust Incremental Bakpropagation Training of STDNN

The backpropagation algorithm performs a forward and backward pass through the network. The forward pass feeds the inputs through the network and finally the output of the neural network model. The backward pass calculates the error derivatives. We formulate the robust backpropagation training algorithm for the extended studentized STDNN network as follows:

Incremental Bakpropagation Algorithm for Studentized STDNN

Initialization : Data $\{(x_t, y_t)\}(t = 1..T)$, initial weights set to small values, learning rate $\eta = 0.1$, moving average period p =14

First Phase: Perform training of the *mean network part* only by BackPropagation

Second Phase: Perform training of the *variance network* only *by* BackPropagation

Third Phase: Perform simultaneous training of both network parts

Repeat

For each training example (x_t, y_t) do

i) perform a forward pass through the *mean network*:

$$f(x_t, w_t) = \sum_{j=1}^{H} w_j \, \sigma\left(\sum_{i=1}^{d} w_{ij} z_{t-i}\right), \text{ where: } z_{t-i} = \delta(x_{t-i}, p)$$

perform a forward pass through the *mean network*:

$$s_t \equiv \sigma_t^2 = \varepsilon\left(\sum_{i=1}^{d} v_j \, \sigma\left(\sum_{i=1}^{d} v_{ij} z_{t-i}\right)\right), \text{ where: } z_{t-i} = \delta(x_{t-i}, p)$$

ii) perform a backward pass through the *mean network* and compute:

$$\frac{\partial E_t}{\partial f_t} = -\frac{(n+1)e_t}{n s_t + e_t^2}$$

$$\Delta w_j = \eta \, \frac{\partial E_t}{\partial f_t} h_{j,} \text{ where: } h_j = \sigma\left(\sum_{i=1}^{d} w_{ij} z_{t-i}\right)$$

$$\Delta w_{ij} = \eta \, \frac{\partial E_t}{\partial f_t} w_j h_j (1 - h_j) z_{t-i}$$

perform a backward pass through the *variance* network:

$$\frac{\partial E_t}{\partial s_t} = \frac{1}{2 s_t^2}\left[s_t^2 - \frac{(n+1)e_t^2}{(n + e_t^2/(n s_t^2))}\right]$$

$$\Delta v_j = \eta \, \frac{\partial E_t}{\partial s_t} s_t^2 u_{j,} \text{ where: } u_j = \sigma\left(\sum_{i=1}^{d} v_{ij} z_{t-i}\right)$$

$$\Delta v_{ij} = \eta \, \frac{\partial E_t}{\partial s_t} s_t^2 v_j u_j (1 - u_j) z_{t-i}$$

iii) update all weights by the computed changes:

$$w_j = w_j + \Delta w_j, \;\; w_{ij} = w_{ij} + \Delta w_{ij}$$

$$v_j = v_j + \Delta v_j, \quad v_{ij} = v_{ij} + \Delta v_{ij}$$

until the termination condition is satisfied

Our research found that the problems of this double STDNN architecture are that it is slow to train and rather unstable. This occurs not only because the two outputs are dependent on each other, but also because the gradient-descent backpropagation training algorithm is slow, that is it features slow convergence. These features make the extended STDNN developed here not very useful for incremental online training applications.

3 ROBUST RECURSIVE ESTIMATION OF STDNN

In this chapter we focus on the development of heavy-tail non-linear network models and corresponding robust incremental learning algorithms for highly accurate modeling of time series. First, we simplify the network architecture, and second develop a robust recursive estimation algorithm for all its parameters and noise hyperparameters based on Kalman filtering [Harvey, 2001], [Haykin, 2001], [Shumway and Stoffer, 2011], [Grewal and Andrews, 2014] and the Expectation Maximization (EM) framework [Dempster, Laird and Rubin, 1977], [McLachlan and Krishnan, 1997]. Although these theoretical developments have no direct practical translation, they provide the theoretical grounding of the methods used in the empirical work carried out at the end of this paper.

The Extended Kalman Filter (EKF) is a sequential inference algorithm that is also highly recommended for training neural networks on time series [Haykin, 2001]. The advantage of EKF is that it provides a second-order training algorithm which leads to better results than simply using the first-order backpropagation algorithm. However, the traditional EKF is sensitive to outliers and a lot of research has been dedicated to robustifying its performance using heavy-tailed noise distributions [Briegel and Tresp, 2000], [Thing and al., 2007], [Särkkä and Hartikainen, 2013]. Although often contemporary variational inference techniques has been applied [Piché et al., 2012], [Särkkä and Hartikainen, 2013] they involve approximating the Studen-*t* density and using its approximation rather than the density directly.

The EKF is used in the framework of the Expectation Maximisation (EM) algorithm [Dempster, Laird and Rubin, 1977], [McLachlan and Krishnan, 1997] for finding estimates of the unknown model parameters given measurement data. The EM helps us to learn all parameters as well as noise hyperparameters simultaneously during training. In order to achieve a fully incremental training of STDNN here we follow the incremental EM [Neal and Hinton, 1998].

This chapter develops a simplified STDNN that keeps only the mean part of the statistical network architecture, while replacing the whole variance part just by a single node. Thus, we still design an STDNN model that infers the mean and variance of the target distribution, in which we do not have a special model for the variance but rather we re-estimate it at each time step with a suitable formula. Overall, the STDNN remains a non-linear model for the mean learnable with a corresponding time varying variance. Next, we formulate a robust filter for faster and more accurate convergence of the network training process. The filter implements a second-order training method, which features faster convergence than the simple gradient-descent backpropagation. The robust filter is elaborated using partial results from chapter 2, namely the studentized error derivatives. These are the studentized error derivatives that allow us to avoid the detrimental effects of outliers on the weight updates.

3.1. EXPECTATION MAXIMIZATION ALGORITHM

The EM algorithm is one of the main operational tools for time series modeling [Shumway and Stoffer, 1982, 2011]. The EM alternates between estimating the unknown weight parameters and the hidden noise hyperparameters. There are two algorithmic components in this algorithm: E-Step and M-Step. The model parameters are computed via Kalman filtering and smoothing during the E-Step. Once the model parameters have been found in the E-Step, their likelihood function is computed next, and finally the hyperparameters are updated during the

maximization step in the M-Step. The EM is a proper algorithm which ensures convergence while searching for the mode of the complete-data likelihood.

The EM algorithm maximizes the complete-data (joint) log likelihood by alternating the two steps, and adapts iteratively the parameters and the hyperparameters until convergence. The complete-data log likelihood function is given by:

$$\log L(w_{1:T}, y_{1:T}|Q, R) = \log \int p(w_{1:T}, y_{1:T}|Q, R)\, dw_{1:T} \qquad (37)$$

E-step: Calculate the expected log-likelihood of the complete data conditioned at the current hyperparameters:

$$E[\![\log p(w_{1:T}, y_{1:T}|Q, R)]\!] \qquad (38)$$

and learn the network weights given the noises. Learning of the weights involves differentiation of the expected log-likelihood with respect to the parameters and solving after equating the first derivative to zero:

$$\widehat{w}_E = arg\max_{w} E[\![\log p(w_{1:T}|y_{1:T}, Q, R)]\!] \qquad (39)$$

Here we are going to use Kalman filtering and smoothing for weights estimation.

M-Step: Calculate the new values of the parameters Q and R (while keeping the weights unchanged) by optimization as follows (ie maximize with respect to the hyperparameters):

$$(Q^{NEW}, R^{NEW}) = arg\max_{Q,R} E[\![\log p(w_{1:T}, y_{1:T}|Q, R)]\!] \qquad (40)$$

We start the iterations with plausible values of the parameters and the hyperparameters.

The EM algorithm is particularly suited for models with latent variables like the STDNN. In our model the weights are the hidden (latent) variables in the dynamic state-space model. The use of the EM requires to interpret STDNN as a dynamic state-space model, and process it using the sequential inference framework.

3.2 SEQUENTIAL BAYESIAN INFERENCE

The task of the recursive (online) nonlinear STDNN estimation is to infer the values of the network weights along with the arrival of the observations. This motivates us to adopt the sequential Bayesian inference framework [Tanizaki, 1996], [Särkkä, 2013] and handle the variables of interest using their probability distributions. The task of Bayesian training is to compute the posterior probability (and its moments) of the weights at each time instant using all past measurements and next updating it to accommodate the new observation.

Given the data the objective is to determine $p(w_t|y_t)$. The solution to this learning problem is found by recursive inference which filters the posterior distribution of the latent weights. This process can be represented probabilistically using the Bayes rule as follows [Tanizaki, 1996]:

$$p(w_t|y_{1:t-1}) = \int p(w_t|w_{t-1})\, p(w_{t-1}|y_{1:t-1}) dw_{t-1} \qquad (41)$$

$$p(w_t|y_{1:t-1}) = C^{-1}p(y_t|w_t)p(w_t|y_{1:t-1}) \quad (42)$$

where the normalizing constant is $C = \int p(y_t|w_t)\, p(w_t|y_{1:t-1})dw_t$, the density $p(w_t|y_{1:t-1})$ is the weights prior (predictive distribution), $p(w_t|y_{1:t})$ is the weights posterior (filtering distribution), and $p(y_t|w_t)$ is the data likelihood.

It should be noted that the weights posterior $p(w_t|y_{1:t})$ cannot be obtained analytically for this nonlinear STDNN model directly. This difficulty however can be alleviated through linearization with the neural network derivatives as will be shown below.

3.2.1 Linear Kalman Filtering

The Kalman filter [Harvey, 2001], [Haykin, 2001], [Durbin and Koopman, 2001], [Shumway and Stoffer, 2011], [Grewal and Andrews, 2014] has been originally developed for parameter estimation in linear models. The Kalman filter is a minimum mean-square (variance) estimator of the state of a linear dynamical system. The Kalman filter for linear models is defined using two equations:

- process equation that defines the evolution of the weights with time;
- measurement equation that defines the observable in terms of the weights.

Formally, a linear dynamical state-space model can be described in the following way:

Dynamic Linear Model

$$w_t = w_{t-1} + q_t \quad q_t \sim N(0, Q) \quad (43)$$

$$y_t = Gw_t + r_t \quad r_t \sim N(0, R) \quad (44)$$

where q_t and r_t are independent, zero-mean Gaussian noises with covariances *Q* and *R*.

The standard Kalman filter provides equations for updating the model weight parameters with the arrival of each next data tuple, so as to accommodate the new information in the parameters. Thus, the model becomes a better descriptor of the data and it evolves sequentially in time. The standard Kalman filter consists of the following equations [Haykin, 2001]:

State-Transition Equation	$\hat{w}_t^- = \hat{w}_{t-1}$	(45)
Error Covariance	$\hat{P}_t^- = \hat{P}_{t-1} + Q$	(46)
Kalman Gain	$K_t = \hat{P}_t^- G\left[G\hat{P}_t^- G^T + R\right]^{-1}$	(47)
State Update	$\hat{w}_t = \hat{w}_t^- + K_t(y_t - f_t)$	(48)
Error Covariance Update	$\hat{P}_t = (I - K_t G)\hat{P}_t^-$	(49)

where *Q* and *R* are the corresponding state and measurement noise covariances.

3.2.2 Derivation of the Kalman Filter

Here we explain these equations with a derivation of the Kalman filter which we make following relevant research [West and Harrison, 1999], [Särkkä, 2013] and expand on it by showing its components in detail. The estimation of the weight posterior via sequential filtering involves two steps: first, time updating during which we calculate the weight prior, and, second, measurement updating during which we compute the posterior using the obtained prior.

The Bayesian filtering framework relies on the assumption that the weights evolution forms a Markov chain [Särkkä, 2013], that is the current weight vector depends only on the last weight

vector and it is independent from anything before that: $p(w_t|w_{1:t-1}, y_{1:t-1}) = p(w_t|w_{t-1})$. That is, the weights distribution depends only on the most recent probability density.

That is why, the time updating equation is:

$$p(w_t|y_{t-1}) = \int p(w_t|w_{t-1})\, p(w_{t-1}|y_{1:t-1}) dw_{t-1} \qquad (50)$$

Calculating the Weight Prior

This predictive distribution is typically considered Gaussian, whose mean and variance are derived as follows:

$$p(w_t|y_{1:t-1}) = N\left(w_t\middle|\hat{w}_t^-, \hat{P}_t^-\right) \qquad (51)$$

where the predicted mean is:

$$\begin{aligned} \hat{w}_t^- &= E[\![w_t|y_{1:t-1}]\!] \qquad (52)\\ &= E[\![w_{t-1} + q_t|y_{1:t-1}]\!] \\ &= E[\![w_{t-1}|y_{1:t-1}]\!] + E[\![q_t|y_{1:t-1}]\!] \\ &= \hat{w}_{t-1} \end{aligned}$$

and the covariance matrix is:

$$\begin{aligned} \hat{P}_t^- &= Cov[\![w_t|y_{1:t-1}]\!] \qquad (53)\\ &= Cov[\![w_{t-1}|y_{1:t-1}]\!] + Cov[\![q_t|y_{1:t-1}]\!] \\ &= \hat{P}_{t-1} + Q \end{aligned}$$

Note also that these moments follow from the properties of the multivariate Gaussian distribution [Von Mises, 1964], [Rasmussen and Williams, 2006].

Calculating the Weight Posterior

The derivation of the weight posterior distribution is facilitated by the well-known lemma from multivariate analysis which states that the product of two Gaussians leads also to a Gaussian distribution, that is we have:

$$p(w_t, y_t|y_{1:t-1}) = p(y_t|w_t)p(w_t|y_{1:t-1}) \qquad (54)$$

In order to find its moments we apply the properties of the normal distribution to this joint density.

The moments of this density can be found as follows [West and Harrison, 1999],[Särkkä, 2013]:

$$p(y_t, w_t|y_{1:t-1}) = N\left(\begin{vmatrix} w_t \\ y_t \end{vmatrix}\middle|(m_t, V_t)\right) \qquad (55)$$

where:

$$m_t = \begin{vmatrix} \hat{w}_t^- \\ f(x_t, \hat{w}_t^-) \end{vmatrix}$$

$$V_t = \begin{vmatrix} \hat{P}_t^- & \hat{P}_t^- \hat{g}_t \\ \hat{P}_t^- \hat{g}_t & \hat{g}_t \hat{P}_t^- \hat{g}_t' + R \end{vmatrix}$$

that is, the predicted mean $\hat{w}_t^-$ is estimated during the time updating step along the predicted covariance matrix $\hat{P}_t^-$.

Therefore, in order to obtain the weight posterior moments we just have to explain how to compute the remaining quantities of interest.

The characteristics of the measurement distribution can be found in an analogous way:

$$p(y_t|y_{1:t-1}) = N\left(y_t|f(x_t, \hat{w}_t^-), \left(\hat{g}_t \hat{P}_t^- \hat{g}_t' + R_t\right)\right) \qquad (56)$$

where the mean and its covariance are computed as follows:

$$E[\![y_t|y_{1:t-1}]\!] = E[\![f(x_t, w_t)]\!] + E[\![r_t|y_{1:t-1}]\!] \qquad (57)$$
$$= f(x_t, \hat{w}_t^-)$$
$$Cov[\![y_t|y_{1:t-1}]\!] = \hat{g}_t Cov[\![w_t|y_{1:t-1}]\!]\hat{g}_t' + Cov[\![r_t|y_{1:t-1}]\!] \qquad (58)$$
$$= \hat{g}_t \hat{P}_t^- \hat{g}_t' + R$$

There remains finally to find the equation for the cross variance off-diagonal component: $Cov[\![w_t, y_t]\!]$ of the posterior covariance matrix:

$$Cov[\![w_t, y_t|y_{1:t-1}]\!] = Cov[\![w_t, f(x_t, w_t) + r_t|y_{1:t-1}]\!]$$
$$= E[\![w_t|y_{1:t-1}]\!]\hat{g}_t + 0$$
$$= \hat{P}_t^- \hat{g}_t \qquad (59)$$

Having thus found the moments of the joint distribution, $p(y_t, w_t|y_{1:t-1})$ one can obtain the conditional $p(w_t|y_{1:t})$ according to a well known lemma for multivariate Gaussians, which states that for the jointly Gaussian variables *w* and *y* with moments:

$$\begin{vmatrix} w \\ y \end{vmatrix} \sim N\left(\begin{vmatrix} \mu_w \\ \mu_y \end{vmatrix}, \begin{vmatrix} A & B \\ B' & C \end{vmatrix}\right) \qquad (60)$$

the conditional $w|y$ is also Gaussian with moments:

$$w|y \sim N\left(\left(\mu_w + BC^{-1}(y - \mu_y)\right), (A - BC^{-1}B')\right)$$

Having these properties applying them to the joint leads to the following posterior:

$$p(w_t|y_{1:t}) = N\left(w_t|\hat{w}_t, \hat{P}_t\right)$$

where the mean vector and the covariance matrix are computable recursively.

Note that here we explained one particular version of the derivation of the Kalman filter for linear models, but other derivations of the Kalman filter are also available [Maybeck, 1979], [Harvey, 2001], [Durbin and Koopman, 2001].

3.2.3 Extended Kalman Filtering of STDNN

The STDNN is actually a kind of state-space model whose weights represent the hidden, unobserved state of the dynamical system that describes the evolution of the given measurements, which can be formulated with a couple of equations as follows:

Process (transition) equation

$$w_t = w_{t-1} + q_t, \qquad q_t \sim N(0, Q) \qquad (61)$$

Measurement equation

$$y_t = f(x_t, w_t) + r_t, \quad r_t \sim N(0, R) \qquad (62)$$

where the noises q_t and r_t are assumed to be Gaussian with unknown variances.

These noises are considered independent (we say that the neural network function $f(x_t, w_t)$ is driven by measurement, that is observational noise).

The STDNN under development is a non-linear temporal model that requires to define an Extended Kalman Filter for it. The EKF is an application of the Kalman filter to nonlinear dynamical models like our STDNN network after linerarizing the network through the output gradient with respect to the states. The equations of the KF are convertible to an EKF for the studied here nonlinear STDNN model. This is possible through linearization of the neural network model using the vector $\hat{g}_t$ of weight derivatives computed by the backprop algorithm.

The gradient $\hat{g}_t$ is a vector of derivatives of the neural network output with respect to each weight $\hat{w}_{ij}^-$, that is:

$$\hat{g}_t = \frac{\partial f(x_t, \hat{w}_t^-)}{\partial \hat{w}_{t,ij}^-}$$

which is evaluated with the backpropagation algorithm. It measures the sensitivity of the network output to local changes in each of the weights, with which linearization of the measurement model is achieved (using a first-order Taylor series expansion).

The Extended Kalman Filter (EKF) operating over the linearized version of the model through derivatives is defined as follows [Shumway and Stoffer, 1982], [Haykin, 2001]:

$$\hat{w}_t^- = \hat{w}_{t-1} \qquad (64)$$

$$\hat{P}_t^- = \hat{P}_{t-1} + Q \qquad (65)$$

$$\hat{g}_t = \frac{\partial f_t(x_t, \hat{w}_t^-)}{\partial x_t} \qquad (66)$$

$$K_t = \hat{P}_t^- \hat{g}_t \left(\hat{g}_t \hat{P}_t^- \hat{g}_t' + R\right)^{-1} \qquad (67)$$

$$\hat{w}_t = \hat{w}_t^- + K_t\left(y_t - f(x_t, \hat{w}_t^-)\right) \qquad (68)$$

$$\hat{P}_t = \hat{P}_t^-\left(I - K_t \widehat{g'}_t\right) \qquad (69)$$

where $\hat{g}_t$ is the estimated gradient of the output error with respect to the weights vector.

One disadvantage of EKF is that because it is based on a local linear approximation, it may not work well on problems with considerable non-linearities. Therefore a research strategy could consist of finding appropriate outlier removal methods so that one can keep only the advantage while reducing the disadvantage.

3.3. ROBUST STDNN FILTERING

The Gaussian Kalman filter however is quite sensitive to moderate and large outliers in the training seres (and even sometime unreliable). Even one large outlier can cause untypical weight updates and lead to distortion in the model. Distortion here means a deviation of the model from the true unknown mapping and its assumed form. Special robustification is

necessary in order to prevent the learning from such undesirable effects on the model by detecting large outliers and decreasing their influence on the weight updating process.

This motivates us to conduct research into formulation of a Kalman filter robust to outliers. One strategy is to use a modified cost function for optimization which limits the effects of outliers in a monotonic fashion. Thus, the impact of outliers can be reduced during training. We propose to use directly the Student-*t* pdf for describing the target distribution (which was introduced in Chapter 2), and elaborate a robust filter according to the approximate conditional mean filtering approach suggested by Masreliez [1978]. Using the long-tail Student-*t* density makes the error less heavily influenced by large outliers.

The formulae leading to outlier resistance are obtained by taking the derivative of the error function which yields the so called score function $\psi(e_t)$, that actually helps to decrease gradually the impact of large errors (caused obviously by outliers). The novelty is in the use of a studentized score function for reducing the effects of the outliers. Actually we already showed our own derivation of this studentized score function in chapter 2. The score function for filter robustification is a partial result from the network gradient-based training equations. What remains is to obtain also the derivative of the score function, and to clarify how to plug this score function into the filter.

The robust filter is formulated as follows:

$$\hat{w}_t^- = \hat{w}_{t-1} \qquad (70)$$

$$\hat{P}_t^- = \hat{P}_{t-1} + Q \qquad (71)$$

$$\hat{g}_t = \frac{\partial f_t(x_t, \hat{w}_t^-)}{\partial x_t} \qquad (72)$$

$$e_t = y_t - f(x_t, \hat{w}_t^-) \qquad (73)$$

$$\hat{w}_t = \hat{w}_t^- + \hat{P}_t^- \hat{g}_t \psi(y_t) \qquad (74)$$

$$\hat{P}_t = \hat{P}_t^- - \hat{P}_t^- \hat{g}_t \psi'(y_t) \hat{g}_t' \hat{P}_t^- \qquad (75)$$

where: $\psi(y_t) = (-\partial \log p_s(y_t|Y_{t-1}) / \partial e_t)(\partial e_t / \partial f_t)$ is the so called score function and $\psi'(y_t) = \partial \psi(e_t)/\partial e_t$ is its derivative.

We already showed a similar derivation of the score in Chapter 2 (where it has been obtained as a partial result necessary for training the STDNN by robust backpropagation), which we repeat here for clarity:

$$\psi(y_t) = -\frac{\partial \log p_s(y_t|Y_{t-1})}{\partial e_t}\frac{\partial e_t}{\partial y_t}$$

$$= \frac{\partial \left[(n+1)\log\left(1+e_t^2/\left(ns_t^2\ \right)\right)/2\right]}{\partial e_t}\frac{\partial e_t}{\partial y_t}$$

$$= \left(\frac{n+1}{2}\right)\left(\frac{1}{1+e_t^2/\left(ns_t^2\ \right)}\right)\left(\frac{2e_t}{ns_t^2}\right)$$

$$= \left(\frac{n+1}{2}\right)\frac{2e_t(ns_t)}{ns_t^2+e_t^2}\frac{1}{ns_t^2}$$

$$= \frac{(n+1)e_t}{ns_t^2 + e_t^2} \qquad (76)$$

where *n* is the degree of freedom parameter of the Student-*t* distribution.

What remains is to find the derivative of this score function. The derivative of the score function is obtained as follows:

$$\psi'(y_t) = \frac{(n+1)(ns_t^2 - e_t^2)}{(ns_t^2 + e_t^2)^2}$$

$$= \frac{(n+1)}{(n + e_t^2/s_t)^2} \frac{(n - e_t^2/s_t^2)^2}{s_t^2} \qquad (77)$$

One can check the validity of our equations by replacing the Student-t assumption with the Gaussian assumption for the observations, and realize that our equation then become identical to these of the Kalman filter. Our robust EKF actually downweights the large residuals, thus reducing their effect when updating the weights.

3.4. THE RTS SMOOTHING ALGORITHM

The sequential learning of TDNN models from time series requires to evaluate the likelihood with the complete data, so after the forward pass we perform a backward smoothing pass from the series end down to the beginning. The sequential estimation process involves filtering during which we estimate the hidden weights (states) given the data arrived up to the current moment, and after that smoothing backwards from the current moment in reverse to the beginning of the series. Thus, improved estimates of the posterior density $p(w_t|y_{1:T})$ using subsequently arrived information, not available during the forward pass are obtained.

After smoothing we obtain improved averaged values of the learned weights. This is because when we filter in forward manner the model weights are learned only from incomplete partial information available only up to the particular moment in time, and when we return backwards and re-evaluate the model we actually update the weights further using the unseen before information (from the future).

In scientific parlance the purpose of Bayesian smoothing is to compute the marginal posterior distribution of the weights after receiving all data. The smoother is conditioned on the complete data, that is on available training data. The smoothing equations are derived from the integral of the following joint distribution [Tanizaki, 1996]:

$$p(w_t|y_{1:T}) = \int p(w_t, w_{t+1}|y_{1:T})\, dw_{t+1} = \int p(w_t|w_{t+1}, y_{1:T}) p(w_{t+1}|y_{1:T})\, dw_{t+1} \qquad (78)$$

which follows from the Markovian property of the hidden states. The resulted smoothed posterior $p(w_t|y_{1:T})$ is assumed here Gaussian.

We consider the Rauch-Tung-Striebel (RTS) type algorithm [Harvey, 2001], [Haykin, 2001], [Durbin and Koopman, 2001], for computing the smoothed weight posterior defined by the following recursive equations [Shumway and Stoffer, 1982], [Ghahramani and Hinton, 1996]:

$$J_{t-1} = \hat{P}_{t-1}\left(\hat{P}_t^-\right)^{-1} \qquad (79)$$

$$\hat{w}_{t-1}^T = \hat{w}_{t-1} + J_{t-1}(\hat{w}_t^T - \hat{w}_t^-) \qquad (80)$$

$$\hat{P}_{t-1}^T = \hat{P}_{t-1} + J_{t-1}\left(\hat{P}_t^T - \hat{P}_t^-\right)J_{t-1}' \qquad (81)$$

where the recursions begin with $\hat{w}_t^T = \hat{w}_T$ and $\hat{P}_t^T = \hat{P}_T$.

One has to compute also the cross-covariances defined as follows [Shumway and Stoffer, 1982], [Ghahramani and Hinton, 1996]:

$$\hat{P}_{t-1,t-2}^T = \hat{P}_{t-1}J_{t-2}' + J_{t-1}\left(\hat{P}_{t,t-1}^T - \hat{P}_{t-1}\right)J_{t-1}' \qquad (82)$$

which starts with the initial condition $\hat{P}_{T,T-1}^T = (I - K_t\hat{g}_t)\hat{P}_{T-1}^{T-1}$.

3.5. INCREMENTAL PARAMETER ESTIMATION

The task of learning all model weight parameters and noise hyperparameters together can be accomplished simultaneously during training using the EM algorithm [Dempster, Laird and Rubin, 1977], [McLachlan and Krishnan, 1997]. After filtering and smoothing of the weights it suggests to compute estimates of the noise hyperparameters. Having specified a modified STDNN with one mean output network there remains to derive equations for finding the measurement noise variance $R_t \equiv \sigma_t^2$ as well as the process noise variance Q. Note here that we are interested in estimating a time-varying output noise R_t so we are going to make an incremental version of the EM algorithm [Neal and Hinton, 1998]. Thus, our STDNN model will still remain a kind of density neural network that is trained in an online manner.

3.5.1 The Complete-Data Likelihood

The EM algorithm searches for the maximum of the cost $E[\![\log p(w_{1:T}, y_{1:T}|Q,R)]\!]$. The algorithm alternates between expectation and maximization steps. The likelihood cost has to be computed with the completed data, so we perform a forward filtering following by a backward smoothing pass. The approximation to the full data density is computed via filtering of the dynamic state-space STDNN model. The likelihood of the complete data, assuming as stated before uncorrelated state and measurement noises, and normal Gaussian weight density $p(w_t|w_{t-1})$ and Student-*t* observation density $p_s(y_t|w_t, R_t) \equiv p_s(y_t|Y_{t-1})$ is:

$$p(w_{1:T}, y_{1:T}|Q, R_t) = p(w_0)\prod_{t=1}^{T} p(w_t|w_{t-1})\prod_{t=1}^{T} p_s(y_t|w_t, R_t) \qquad (83)$$

which is used in the complete-data log likelihood function $\log L(w_{1:T}, y_{1:T}|Q, R_t)$.

The ingredient densities are defined in the following way:

$$p(w_0) = \frac{1}{2\pi^{\frac{m}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left[-\frac{1}{2}(w_0 - \mu)'\Sigma^{-1}(w_0 - \mu)\right] \qquad (84)$$

$$p(w_t|w_{t-1}, Q) = \frac{1}{2\pi^{\frac{m}{2}}|\mathrm{Q}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(w_t - w_{t-1})' Q^{-1}(w_t - w_{t-1})\right] \quad (85)$$

$$p_s(y_t|w_t, R_t) = \frac{\Gamma((n+1)/2)}{\sqrt{\pi n R_t}\Gamma(n/2)}\left(1 + \frac{(y_t - f_t)^2}{nR_t}\right)^{-\frac{(n+1)}{2}} \quad (86)$$

where $R_t \equiv \sigma_t^2$ is the precision, n $(n > 2)$ is the degree of freedom parameter of the Student-t distribution, and $\Gamma(\cdot)$ is the Gamma function.

Taking the logarithm of this complete-data likelihood we obtain the logarithmic joint posterior:

$$\log p(w_{1:T}, y_{1:T}|Q,R) = -\frac{T}{2}\log|Q| - \sum_{t=1}^{T}\left[\frac{1}{2}(w_t - w_{t-1})'Q^{-1}(w_t - w_{t-1})\right] - \sum_{t=1}^{T}\left[\frac{1}{2}\log R_t + \frac{n+1}{2}\log\left(1 + \frac{(y_t - f_t)^2}{nR_t}\right)\right] \quad (87)$$

where the terms from the initialization and the terms from the degree of freedom parameter are omitted for clarity (since they do not affect the derivations of the equations for training of the hyperparameters demonstrated below).

Taking the expectation of this logarithm of the complete-data likelihood yields:

$$E[\![\log p(w_{1:T}, y_{1:T}|Q,R_t)]\!] = -\frac{1}{2}\sum_{t=1}^{T}\log R_t - \sum_{t=1}^{T}\left[E\left[\!\!\left[\frac{n+1}{2}\log\left(1 + \frac{(y_t - f_t)^2}{nR_t}\right)\right]\!\!\right]\right] - \frac{T}{2}\log Q - \sum_{t=1}^{T}\left[\frac{1}{2}E[\![(w_t - w_{t-1})'Q^{-1}(w_t - w_{t-1})]\!]\right] \quad (88)$$

Next, we take the derivative of this expected log-likelihood equate it to zero, and solve it for hyperparameters Q and R_t.

3.5.2 Estimating the Process Noise Variance

The process noise variance hyperparameter is estimated in batch (offline) manner. After differentiation with respect to the noise hyperparameter Q we get:

$$\frac{\partial E[\![\log p(w_{1:T}, y_{1:T}|Q,R_t)]\!]}{\partial Q} = Q\frac{(T-1)}{2} - \frac{1}{2}\left(A - 2B^{T'} + C^{T'}\right) \quad (89)$$

$$Q = \frac{1}{(T-1)}\left(A - BC^{-1}, B^{T'}\right) \quad (90)$$

where we have used the following matrices:

$$C = \sum_{t=1}^{T}[P_{t+1}^{T} + w_{t-1}^{T}(w_{t-1}^{T})'] \quad (91)$$

$$B = \sum_{t=1}^{T}\left[P_{t|t+1}^{T} + w_t^{T}(w_{t-1}^{T})'\right] \quad (92)$$

$$A = \sum_{t=1}^{T}[P_t^{T} + w_t^{T}(w_t^{T})'] \quad (93)$$

where the prime symbol $'$ denotes the transpose of a vector.

3.5.3 Estimating the Measurement Noise Variance

Let assume that the overall measurement variance is $R = (1/T)\sum_{t=1}^{T} R_t$. After differentiation with respect to the noise hyperparameter R we get:

$$\frac{\partial E[\![\log p(w_{1:T}, y_{1:T}|Q, R_t)]\!]}{\partial R} =$$

$$= -\frac{1}{2}\sum_{t=1}^{T}\frac{\partial \log R_t}{\partial R_t} - \sum_{t=1}^{T}\frac{\partial E\left[\!\left[(n+1)\log\frac{(1+e_t^2/(nR_t))}{2}\right]\!\right]}{\partial R_t}$$

$$= \sum_{t=1}^{T}\left[-\frac{1}{2R_t} + \frac{1}{2R_t^2}\frac{(n+1)E[\![e_t^2]\!]}{(n+e_t^2/R_t)}\right]$$

$$= \sum_{t=1}^{T}\left(-\frac{1}{2R_t^2}\left[R_t + \frac{(n+1)E[\![e_t^2]\!]}{(n+e_t^2/R_t)}\right]\right) \qquad (94)$$

After equating to zero and solving we arrive at the following equation:

$$R = \frac{1}{T}\sum_{t=1}^{T}\frac{(n+1)}{(n+e_t^2/R_t)}\left[\hat{g}_t\hat{P}_t^{-}\hat{g}_t' + \left(y_t - f(x_t, w_t)\right)^2\right] \qquad (95)$$

which obviously is tailored to our STDNN neural network using the weight gradient $\hat{g}$.

This formula clearly shows that the error is actually downweighted by the multiplier $(n+1)/(n+e_t^2/R_t)$. Thus, large outliers are treated in such a way that they do not have practically any effect on the updating of the weights, that is large outliers are effectively weighted down. A similar formula has been already suggested for offline estimation of neural networks [Briegel and Tresp, 2000], but here however we are willing to develop an online estimation algorithm for temporal networks.

Since we aim at design of a fully incremental training algorithm for the STDNN, we proceed by implementing an iterative version of this formula for recursive estimation of the measurement noise variance hyperparameter R_t at every time step.

$$R_t = \frac{1}{t}\left(\sum_{i=1}^{t-1} R_i + \frac{(n+1)}{(n+e_t^2/R_{t-1})}\left[\hat{g}_t\hat{P}_t^{-}\hat{g}_t' + \left(y_t - f(x_t, w_t)\right)^2\right]\right) \qquad (96)$$

where the current noise R_t is approximated inside the formula roughly by the last value R_{t-1}.

It should be noted that this iterative recalculation of the output noise can be made also using a forgetting rate as often done in the field of signal processing.

3.6. ALTERNATIVE ROBUST FILTERS

We are interested in handling outliers in order to process accurately real-time financial data are contaminated by noise and spikes. Our intention is to develop robust training procedures that identify correct models without, that is with less influence by the noise. Proper handling of noise allows to identify more precisely the underlying model. This is especially useful for STDNN which are highly nonlinear models sensitive to noise.

Recent previous research [Briegel and Tresp, 2000], [Thing and al., 2007], [Särkkä and Hartikainen, 2013] investigated the use of heavy-tailed distributions that help to describe adequately noisy data when training online on time series. We are going further and employ a more general but still very flexible noise model by representing the heavy-tail Student-t distribution as a mixture of Gaussians [McLaghlan and Peel, 1997]. This enables us to derive training equations for the TDNN weights and the parameters of the noise.

Currently state-space modeling using Student-*t* Noise distributions is a hot topic [Solin & Särkkä, 2015]. Moreover, our crucial problem in modeling practical financial time series is to handle effectively the outlier so as to achieve accurate results and forecasts.

Robust Sequential Ttaining of MLP

This is one of the first approaches to handle properly outliers through the use of the Student-t distribution when training sequentially MLP neural networks was proposed by [Briegel and Tresp, 2000]. The authors replace the Gaussian noise assumption with the Student-t distribution in the formulation of the error model, and arrive at concrete formulae for learning the network weights as well as the distributional parameters degrees of freedom and scale factor. It should be noted that the formulae that they have derived affect the weights without much distortion when an outlier occurs in the data, and so contribute for careful training and stable learning performance. Otherwise, it is well known that additive measurement outliers lead to instability in sequential learning especially from time series data.

Since even the popular Kalman filter is not robust to outliers, the authors plug the heavy-tail noise model to elaborate an outlier robust Fisher scoring learning algorithm. This is an algorithm that directly infers the posterior mean weights (and their covariances) at every time step, which is especially developed for sequential estimation. Actually the work of Briegel and Tresp [2000] develops a version of the Fisher scoring algorithm for robust sequential estimation of feed-forward neural networks on time series. One criticism, however, to their approach is that it is rather computationally expensive because of the use of matrix inversions, which prevented it from widespread use, in other words it seems suitable mainly for very small size impractical networks.

Outlier Robust Kalman Filtering

The research conducted by Ting [Ting and al., 2007], proposed yet another outlier robust Kalman filter equipped with an EM algorithm for re-estimation of the hyperparameters. In this sense it also performs automatic computation of the noise parameters in the dynamic model. It

is faster than the approach of Briegel and Tresp [2000] because it performs recursive computation of the inverse weight covariance matrix during robust sequential filtering. However, the approach of [Ting and al. [2007] does not apply a heavy-tailed distribution for the output noise, rather it elaborates a version of weighted regression, therefore it does not use a properly defined probabilistic dynamic model that can inherently cope with eventual outliers in the data. This remains a stable strategy for removing outliers by weighting them with respect to summary data statistics and handling them in an efficient manner. Although these weights associated with the data points are given a flat Gamma distribution to account for various data characteristics, they still remain a good but not entirely probabilistic approach to weighted regression which makes the Kalman filter useful for robust sequential processing of time series.

The remaining problem in this research is how to control the magnitude of the weights so as to impact deliberately the effect from the undesired and harmful outliers.

Variational Bayesian Filtering

Our developments will be similar to the most recent approach using the t-Student distribution in place of the Gaussian for robust sequential estimation of dynamic models called filtering by variational Bayesian approximations [Piché et al., 2012], [Särkkä and Hartikainen, 2013]. Its distinguishing characteristic is that it approximates the weights posterior distribution (including the mean and the variance) at each step using the variational Bayesian technique. It obtains the posterior in an efficient manner using factoring through simpler distributions for all of the model parameters. The possibility to deal with outliers comes from the use of an independent Inverse-Gamma representation of the Gaussian noise distribution.

The authors derive corresponding formulae for the weights as well as for the hyperparameters of the probabilistic model. However in their adaptive filtering algorithm the effect of the outliers seem still not entirely controllable as their magnitude can not be tackled with completely in an automatic way. In this sense their approach seems to be for online learning only without full capabilities for outlier accommodation. We are going to continue the research on this approach and extend it to accommodate more accurately the outliers, thus making a really robust approach to sequential filtering for practical real-world time series.

4 MODELING FINANCIAL TIME SERIES

4.1. TIME SERIES PREDICTION

Many potential applications of neural networks involve processing of data arriving sequentially in time. The goal is to predict the next value a short time ahead into the future. Techniques based on feed-forward networks can be applied directly to such problems provided the data is appropriately pre-processed first.

Consider for simplicity a single variable *x*. One common approach is to sample *x* at regular intervals to generate a series of discrete values x_{t-1}, x_t, x_{t+1} and so on. We can take a set of d such values $x_{t-d+1}, \dots, x_t$ and use the next value x_{t+1} as the target for the output of the network.

Once the network has been trained, it can be given a set of observed values $x_{t-d+1}, \dots, x_t$ and used to make prediction for x_{t+1} . This is called one-step-ahead prediction. If the predictions themselves are passed again to the inputs of the network, then predictions can be made at future points x_{t+2} and so on. This is called multi-step ahead prediction, and is typically characterized by a rapidly increasing divergence between the predicted and observed values, due to the accumulation of errors. One drawback with this technique is the need to choose the time increment between successive inputs, and this may require some empirical optimization. Another problem is that the time series may show an underlying trend, such as a steadily increasing value, with more complex structure superimposed.

The predictability of a time series can be studied from a static perspective using statistical tests such as [Nikolaev and Iba, 2006]:

1) tests for autoregressive behavior- applying autocorrelation tests, like the Box-Pierce Q-statistic and the Ljung-Box Q-statistic, to the given series helps to realize whether the future series data depend on the past, that is such tests provide statistically significant evidence that the series is predictable;

2) tests for nonstationarity- such tests allow us to find out whether there is a nonstationary component in a series, for example using unit root tests such as the Dickey-Fuller (DF) test;

3) tests for random walk behavior- in order to find out if a series is uncorrelated, measurements can be made to see whether the series contains random increments. Random behavior in a time series can be examined using variance-ratio tests for deviations from random walk;

4) tests for nonlinearity- motivation for discarding linear models can be obtained by tests for nonlinearity that check whether the data contain evidence for nonlinear signal dependence. This can be done using the Brock-Dechert-Sheinkman (BDS) test.

The predictability of a time series can also be investigated from a dynamical system perspective [Abarbanel, 1996] using invariant series properties, such as the Lyapunov spectrum and the correlation dimension. These dynamic properties characterize the sensitivity of the series to the initial conditions and they quantify the uncertainty about the future series behavior. Positive Lyapunov exponents and noninteger correlation dimension indicate that the series is chaotic. When the time series is chaotic it is difficult to infer stable forecast from different starting points. All these statistical and dynamic tests for time series are available from general-purpose and specialized software tools.

In this thesis we adopt several statistical and econometric measures to estimate the performance of the studied robust TDNN neural networks [Miazhynskaia et al., 2005].

4.2. PREFORMANCE MEASURES

Normalized Mean Squared Error (NMSE)

The Normalized Mean Square Error (NMSE) is defined as follows:

$$NMSE = \sqrt{\frac{\sum_{t=1}^{T}\left(y_t - f(x_t, \hat{w}_t)\right)^2}{\sum_{t=1}^{T}(y_t - y_{t-1})^2}} \qquad (97)$$

which relates the mean square error of the model to the mean square error of the naïve model.

Normalized Mean Absolute Error (NMAE)

The Normalized Mean Absolute Error (NMAE) is defined as follows:

$$NMAE = \frac{\sum_{t=1}^{T}|r_t^2 - \hat{r}_t^2|}{\sum_{t=1}^{T}|r_t^2 - r_{t-1}^2|} \qquad (98)$$

where the return is $r_t = y_t - y_{t-1}$, and respectively $\hat{r}_t = f(x_t, \hat{w}_t) - f(x_{t-1}, \hat{w}_{t-1})$.

Hits (HR)

The ability of the model to predict increases and decreases in returns can be investigated with the help of the Hit Rate (HR) defined as follows:

$$HR = \frac{1}{T}\sum_{t=1}^{T}\Theta_t \qquad (99)$$

where:

$$\Theta_t = \begin{cases} 1, & (f^2(x_t, \hat{w}_t) - r_{t-1}^2)(r_t^2 - r_{t-1}^2) \geq \emptyset \\ \emptyset, & otherwise \end{cases}$$

which is a measure of how often the model gives the correct direction of change of volatility. $HR \in (0, 1)$, where a value of 0.5 indicates that the model is not better than a random predictor generating a random sequence of up and down moves with equal probability.

Weighted Hit Rate (WHR)

The Weighted Hit Rate (WHR) is defined as follows:

$$WHR = \frac{\sum_{t=1}^{T}[(f^2(x_t, \hat{w}_t) - r_{t-1}^2)(r_t^2 - r_{t-1}^2)]|r_t^2 - r_{t-1}^2|}{\sum_{t=1}^{T}|r_t^2 - r_{t-1}^2|} \qquad (100)$$

4.3. REAL-WORLD FINANCIAL SERIES: The *S&P500* Index

The main practical objective in this research is to investigate the usefulness of robust STDNN for analysis of some properly preprocessed financial time series. More precisely, the intention is to use S&P500 price series so as to generate useful trading signals. The price series is going to be converted into indicator series (RSI etc.), and then the STDNN will be applied to forecast the future evolution of the indicator series. We are going to study whether STDNN can be useful for the prediction of indicator series, obtained from prices, for efficient trading.

Below we take as an example, the evolution of the daily closing prices of the US stock-market index *Standard & Poors* 500 (SP500) over a period of one month. It is extracted from Yahoo finance and will serve as dataset for the illustrations presented in this report.

There are several advantages in our choice of the S&P500 index series:

-data exist since 1952;
-although the composition of the index is not known at any particular point in time data about prices are widely available for this benchmark index;
-because the index is composed of the 500 major US corporates it reflects the broader stock-market and is less sensible to idiosyncratic movements of particular stocks;
-it is a widely used index with high liquidity.

We are going to process daily closing S&P500 prices because:

-it is common in financial research
-it is necessary for me as a practical trader to generate trading daily signals.

This research will deal with daily data, that is closing values of the S&P500 index.

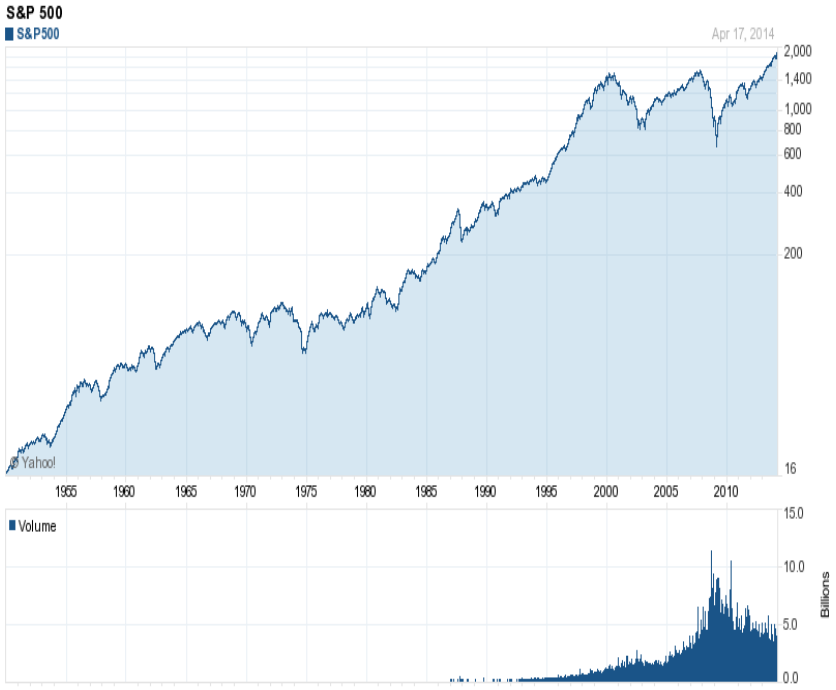


Figure 2: Daily closing prices of SP500 index recorded from 1952 to 2014.

(source: `http://finance.yahoo.com/`)

4.4 TECHNICAL ANALYSIS FOR TRADING

Nowadays computerized trading uses technical indicators to assist for finding and predicting trends in financial time series of prices [Kaufman, 1998]. The main reason for this is the common belief that although prices are unpredictable (theoretically it is considered that they follow a random walk, like a geometric Browian motion) we can identify trends in them. That is why, here we will use technical indicators not only as inputs, but also to try to predict their future evolution. The objective is to help the automatic generation of trading signals (i.e. we do not forecast the evolution of prices directly).

The popular indicators that we are going to use are: the RSI, the Williams%*K*, the DPO [Murphy, 1999], [Brown, 2012], [Kaufman, 2013], [Pring, 2014]. Previous research has found that these are among the fastest oscillators that react fast to the changing trend in financial services series of prices. This is important because they help a trader to react fast in a timely manner and are therefore very useful. Because these indicators are : (a) widely used and (b) are synergetic (and ideally should be used together) as one is better for detecting reversals and the other for detecting momentum.

The problem we find with their application to SP500 is that they are wildly fluctuating and contain a lot of noise that prevent their useful prediction. We intend to use TDNN to smooth them as much as possible in order to facilitate their accurate modeling, and so to make them useful for decision modeling.

Relative Strength Index (RSI)

Our research will focus on modeling the evolution of the RSI index [Murphy, 1999]. This is a well-known and widely accepted momentum indicator which allows us to generate trading signals. However the problem is that any RSI series is usually quite noisy (because of the inherent noise in price series) which makes it less useful for successful trading.

We are going to model and smooth the RSI curve in order to generate useful trading signals (coming directly from the network output). The usefulness of this approach is that we will forecast the future evolution of the RSI indicator and so will facilitate the generation of trading signals. Our objective using a robust filtering technique to smooth the RSI series which will be modeled by a dynamic TDNN model. That is, with the TDNN we intend to build a flexible non-linear representation of the RSI series, which will be further smoothed through robust filtering. Our approach will be applied to S&P500 series using TDNN trained by an EKF following the continued interest in filtering algorithms for financial time series.

RSI Definition. The RSI is a widespread indicator of momentum and of possibility of reversal. The definition of RSI can be found in any standard book on Technical analysis although it may come in many varieties, with as common denominator the ratio of days up versus days down. If we define the average gain over a period by G and the average losses by L, then the basic formula for RSI is:

$$RSI_t = 100 - 100/(1 + \frac{G(x_t)}{L(x_t)})) \qquad (101)$$

where x_t $(1 \leq t \leq T)$ denotes the price at time t.

The RSI is a price-following oscillator that ranges between 0 and 100. A popular method of analyzing the RSI is to look for a divergence in which the security is making a new high, but the RSI is failing to surpass its previous high. This divergence is an indication of an impending reversal. When the RSI then turns down and falls below its most recent trough, it is said to have

completed a "failure swing." The failure swing is considered a confirmation of the impending reversal. Prices usually correct and move in the direction of the RSI.

One problem we have found when applied to real S&P500 prices is that these gains and losses are actually fluctuating and lead to a lot of noise in the RSI, giving rise to unstable signals. That is why, we intend to smooth the RSI through a TDNN. Our preliminary research found that on daily S&P500 price an ideal timeframe of 5 days. Obviously the fewer the days of calculation the more volatile the index becomes.

RSI Trading Strategies. Out of the many possible and practiced trading strategies we chose to focus here on a strategy that is relative simple but helps to illustrate the significant potential contribution of TDNN in generate trading signals when combined with technical indicators.

An empirical rule of thumb in momentum trading is that when a "short" (say 5-session) Moving Average crosses with a "long" one (say 14 –session) this is an indication of an upward trend. Moreover when we choose a "low RSI" area, say below 30, we can relatively safely claim that the issue (here the S&P500) is oversold, as explained above.

The combination of the above two approaches would generate useful trading signals in periods where the market has fallen and is on it is bouncing back.

Williams %K Oscillator

In technical analysis, the Williams %*K* is a momentum oscillator measuring overbought and oversold levels [Murphy, 1999]. It is used to determine market entry and exit points. The Williams %*K* produces values from 0 to -100, a reading over 80 usually indicates a stock is oversold, while readings below 20 suggests a stock is overbought.

Williams%*K*, is a technical analysis oscillator showing the current closing price in relation to the high and low of the past *N* days (for a given period *N*). Its purpose is to tell whether a stock or commodity market is trading near the high or the low, or somewhere in between, of its recent trading range.

$$\%WK_t = 100\ x\ \frac{max(x_t, N) - x_t}{max(x_t, N) - min(x_t, N)} \qquad (102)$$

where x_t is the closing price of the period, $max(x_t, N)$ is the highest closing price of the period, $min(x_t, N)$ is the lowest closing price of the period *N*.

The oscillator is on a negative scale, from −100 (lowest) up to 0 (highest), inverse of the more common 0 to 100 scale found in many technical analysis oscillators. A value of −100 means the close today was the lowest low of the past N days, and 0 means today's close was the highest high of the past N days.

The Williams %*K* provides insights into the weakness or strength of a stock. It's used in various capacities including overbought/oversold levels, momentum confirmations and providing trade signals. To use the Williams %*K* effectively we must understand how it works, its trading applications, as well as its strengths and limitations.

Definition of Williams%K. The %*K* is very similar to the Stochastic Oscillator. The only difference between the two indicators is how they're scaled. The %*K* fluctuates between 0 and

-100, while the Stochastic moves between 0 and 100. The Stochastic also has a moving average applied to it so it can be used for "crossover" signals. The Williams %*K* only has one line by default, although a moving average can be applied to it to give all the functionality of the Stochastic. The indicator was developed by Larry Williams, and shows how the current price compares to the highest price over the look back period. Typically the look back is 14 periods; on a weekly chart that is 14 weeks, on an hourly chart 14 hours.

The most common use for the Williams %*K* is for overbought and oversold readings and momentum confirmations and failures. A security is overbought when the indicator is above -20, and the security is oversold if the indicator is below -80. The labels are misleading, overbought doesn't necessarily mean the price is going to drop soon, and oversold doesn't mean the price is due for a rally. Prices always reverse at some point, but the overbought and oversold don't tell us when this will occur. Overbought simply means the price is trading near the top of the 14 day range. Oversold means the price is trading near the bottom of the 14 range.

Traders do use overbought and oversold levels to monitor reversals though. If the indicator is overbought (above -20) and then falls below -50, traders take this as a sign that the price is moving lower. If the price was oversold (below -80) and rallies above -50 traders take this is a sign the price is moving higher. False signals or late signals occur frequently if these signals are traded unfiltered. Use the price trend to filter the signals.

Detrended Price Oscillator (DPO)

DPO is an oscillator that strips out price trends in an effort to estimate the length of price cycles from peak to peak, or trough to trough. Unlike other oscillators, such as the Stochastic or MACD, detrended price is not a momentum indicator. It highlights peaks and troughs in price, which are used to estimate entry and exit points in line with the historical cycle. The detrended price oscillator formula considered here is:

$$DPO(x_t, p) = x_t - MA(x_t, p) \qquad (103)$$

where *MA* denotes moving average with period p: $MA(x_t, p) = \sum_{i=1}^{p} x_{t-i}$.

The cycles are created because the indicator is displaced back in time. The chart below shows the indicator does not appear at the far right of the chart, and is therefore not a real-time indicator. The historical peaks and troughs in the indicator provide approximate windows of time when it is favorable to look for entries and exits, based on other indicators or strategies.

5. EXPERIMENTAL SECTION

Study of day trading is still in its early stages. As the authors have the honesty to admit: "The choice of the 33 variables used in this study was based on experts knowledge and supported by a vast literature in the area. However, we recognize this approach is adhoc, and the literature is certainly missing more systematic ways to address this specific feature selection problem. As a future research direction, we plan to study the effect of a number of indicators in the network learning, as well as the relevant amount of historical data that should be considered in the inputs of the network (i.e. information regarding the previous 1, 2, 5 or 10 days, for instance)." [Martinez et al., 2014]

The theoretical part of this thesis can be understood as a toolbox for applications in the domain of financial prediction.

There is a reason why all the above methods are individually taken not good enough: they simply don't work well enough, at least taken individually.

Though by combining some of those methods, and adapt them, they can nevertheless synergetically produce limited (but above average) results in a particular context.

NN: time series and classification

Daytrading is not enough studied.

Usually professionals observe the opening of the session and intervene towards the end of the session.

BACKTESTING vs REAL-TIME TESTING

A characteristic of modern quantitative research in finance is that Back-testing has not delivered its promises. Systematically models that have been back tested with past data underperform when confronted with real time data.

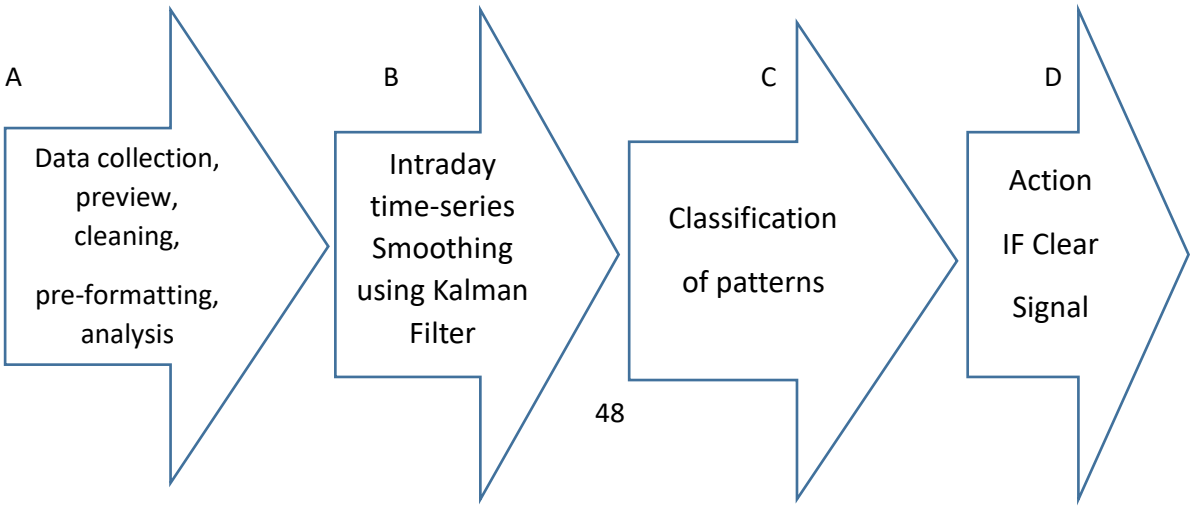
It has now become standard practice to test models with real time data, as this has now become possible with the availability of intraday data, diffused mainly through market participants.

JUDGMENTAL vs "QUANT" the best of two worlds: aim for a decision-aid system not for an automatic system. Moreover, only strong signals should be traded.

5.1. METHODOLOGY OF OUR EMPIRICAL WORK

The methodology followed here consists of tracing the workflow of a typical day trader and proposing a decision-aid system grounded in theory exposed earlier in this study.

Here an integrated process will be proposed structured around four pillars.



Intraday data, smoothing , then classification, then test for forecasting

5.2. STAGE A: DATA

Stage A is an often disregarded part in modelling.

The least-square line is produced here.

A New York Stock Exchange trading day lasts 6h30 (from 14h30 to 21h00) that is 390 minutes, which give 390 data points per day. Multiplied by 126 trading days, that gives 49 920 data points that have to be inspected one by one. This put un upper limit to this time consuming task.

5.3. STAGE B: SMOOTHING SERIES

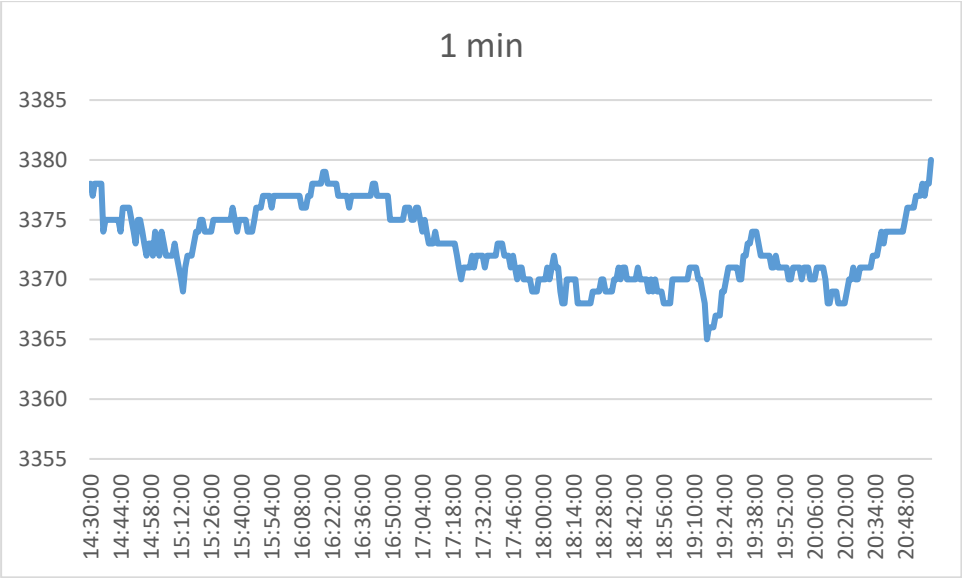
We are going to use intraday data to examine the possibility to forecast intraday data

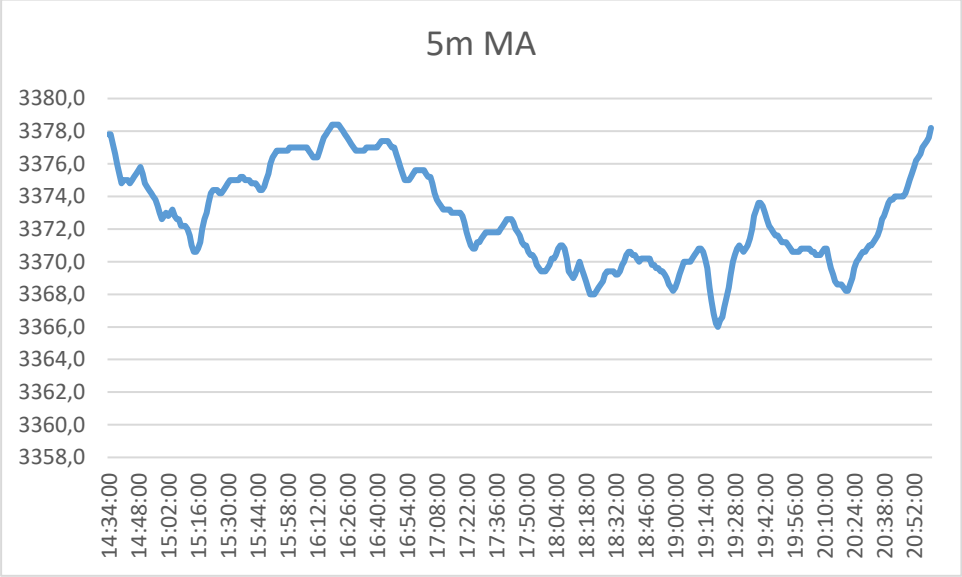
KALMAN FILTER MODEL FOR INTRADAY DATA

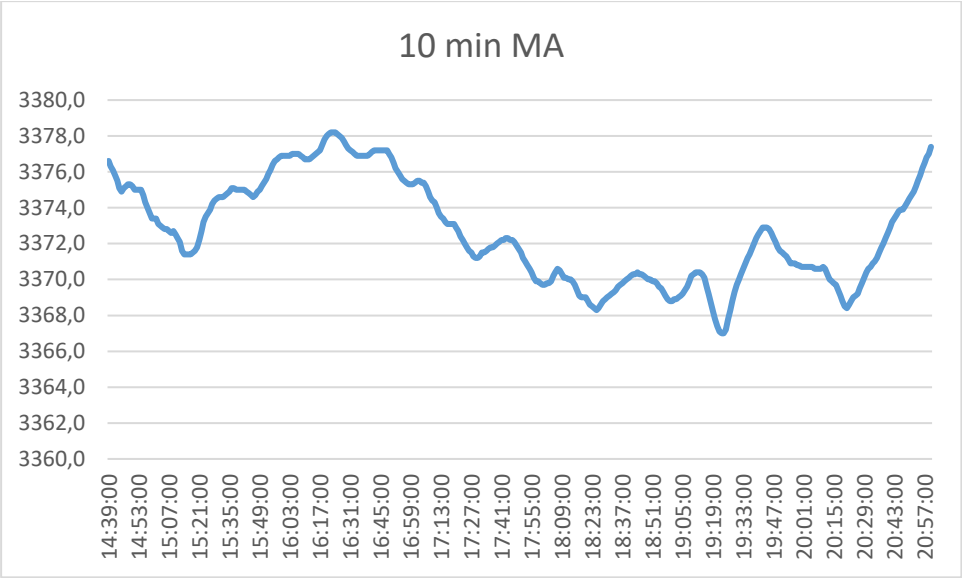
Below we propose an intraday trading model inspired from Kalman Filter.

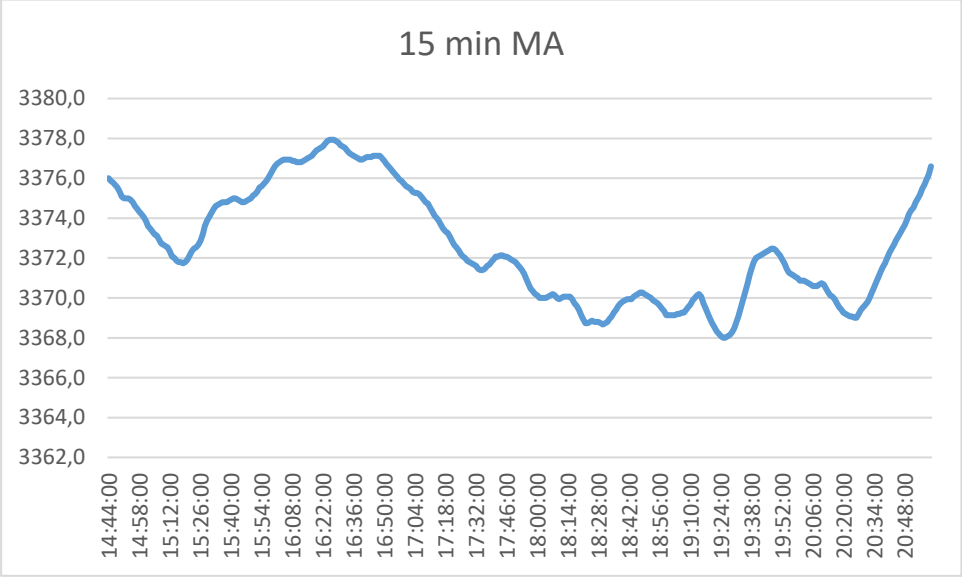
The model is based on intraday data (1 minute to 30 minute) of the SP500 for the day of the 14th of February 2020.

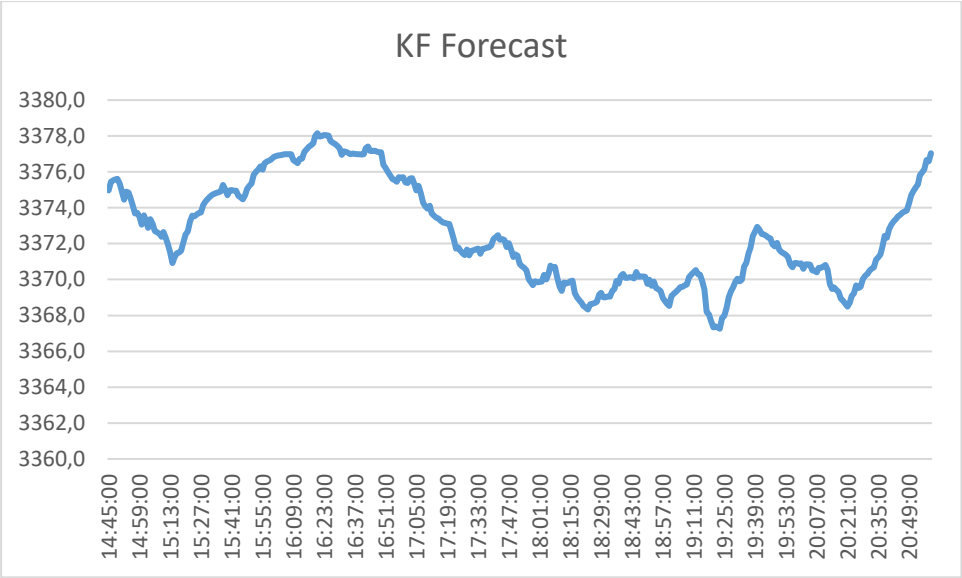
The purpose of this model is to provide forecasts that will be used for trading, every 30 minutes in a typical trading day for the SP500, starting at 14h30 UK time and closing at 21h00


1 min
3385
3380
3375
3370
3365
3360
3355
14:30:00
14:44:00
14:58:00
15:12:00
15:26:00
15:40:00
15:54:00
16:08:00
16:22:00
16:36:00
16:50:00
17:04:00
17:18:00
17:32:00
17:46:00
18:00:00
18:14:00
18:28:00
18:42:00
18:56:00
19:10:00
19:24:00
19:38:00
19:52:00
20:06:00
20:20:00
20:34:00
20:48:00


5m MA
3380,0
3378,0
3376,0
3374,0
3372,0
3370,0
3368,0
3366,0
3364,0
3362,0
3360,0
3358,0
14:34:00
14:48:00
15:02:00
15:16:00
15:30:00
15:44:00
15:58:00
16:12:00
16:26:00
16:40:00
16:54:00
17:08:00
17:22:00
17:36:00
17:50:00
18:04:00
18:18:00
18:32:00
18:46:00
19:00:00
19:14:00
19:28:00
19:42:00
19:56:00
20:10:00
20:24:00
20:38:00
20:52:00


10 min MA
3380,0
3378,0
3376,0
3374,0
3372,0
3370,0
3368,0
3366,0
3364,0
3362,0
3360,0
14:39:00
14:53:00
15:07:00
15:21:00
15:35:00
15:49:00
16:03:00
16:17:00
16:31:00
16:45:00
16:59:00
17:13:00
17:27:00
17:41:00
17:55:00
18:09:00
18:23:00
18:37:00
18:51:00
19:05:00
19:19:00
19:33:00
19:47:00
20:01:00
20:15:00
20:29:00
20:43:00
20:57:00


15 min MA
3380,0
3378,0
3376,0
3374,0
3372,0
3370,0
3368,0
3366,0
3364,0
3362,0
14:44:00
14:58:00
15:12:00
15:26:00
15:40:00
15:54:00
16:08:00
16:22:00
16:36:00
16:50:00
17:04:00
17:18:00
17:32:00
17:46:00
18:00:00
18:14:00
18:28:00
18:42:00
18:56:00
19:10:00
19:24:00
19:38:00
19:52:00
20:06:00
20:20:00
20:34:00
20:48:00



KF for smoothing

5.4. STAGE C: PATTERN CLASSIFICATION FOR DECISION-AID

Here we are the core of our topic

A question of critical importance both for the practitioner and for the student of markets is to detect how a stock price is behaving at a certain point (and its neighborhood) in time, namely is it “trending” or is it bound-ranged.

If the market seems to follow a trend, upwards or downwards, and if the trend seems well established it is then highly likely that the trader will follow that trend (“jump on the trend”, ie “buy” – when the trend seems ascending). On the other hand if the trader believes that the market is “ranging”, ie oscillating, with more or less regularity, then it is more profitable to trade very short term (minutes or hours at most) entering and exiting when the price seems to be near the high/low of the session - provided of course that the patterns present some regularity.

Let us present a decision tree of the patterns of a trading day, and define each term:

#TRENDING: when tested at regular intervals P2>P1, at t2>t1. On the whole the price must be clearly heading to a direction:

-TRENDING / Upwards: that is an overall ascending trend

¤ TRENDING / Upwards / Trend Confirmation: the trend is confirmed and more rarely accelerated

¤ TRENDING / Upwards / Trend Reversal: the trend is reversed

-TRENDING / Downwards: an overall descending trend

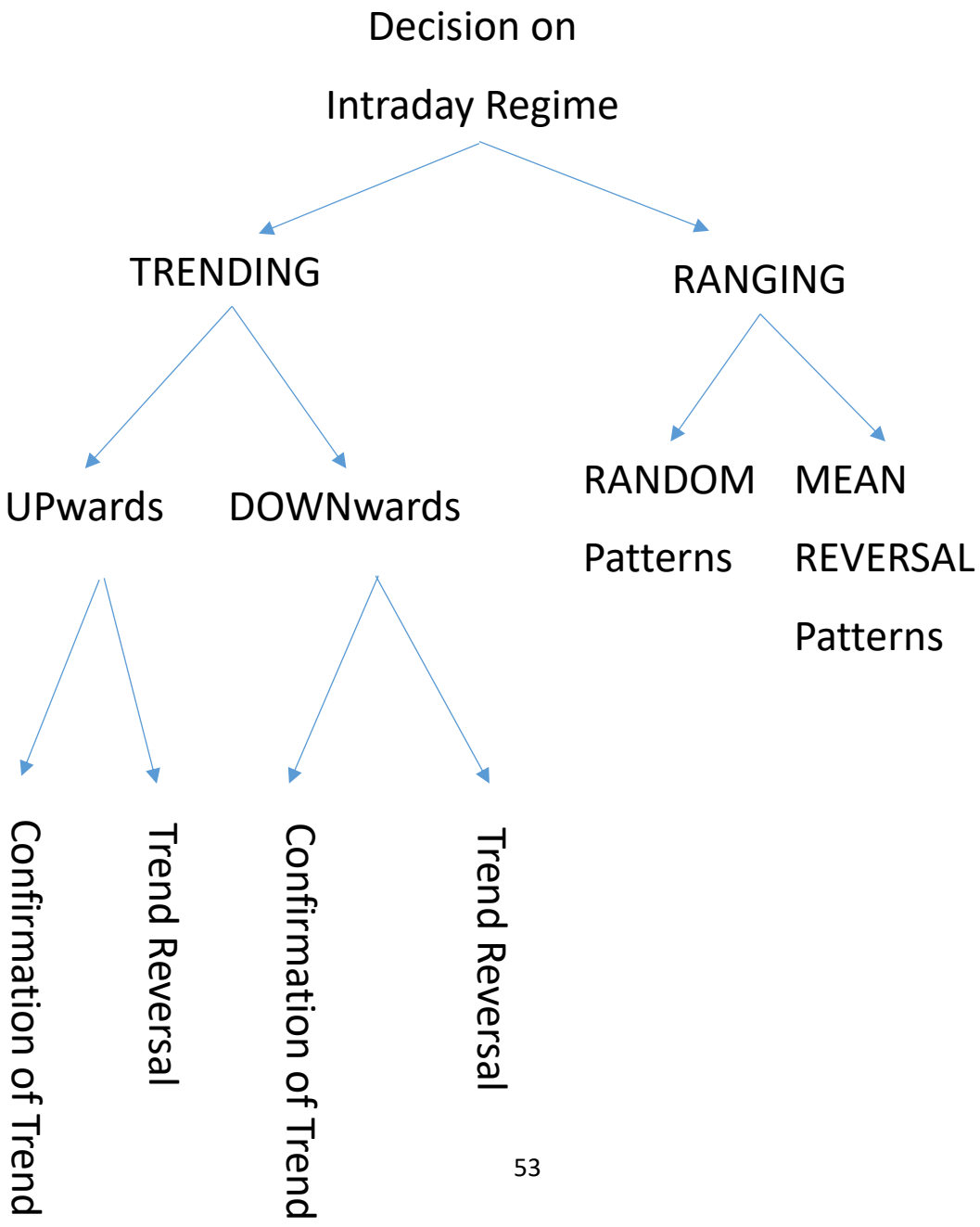
¤ TRENDING / Downwards/ Trend Confirmation

¤ TRENDING / Downwards / Trend Reversal

#RANGING: the price is range bound. Formally, if the price is not trending then it is ranging.

-RANGING / Mean Reversal Patterns

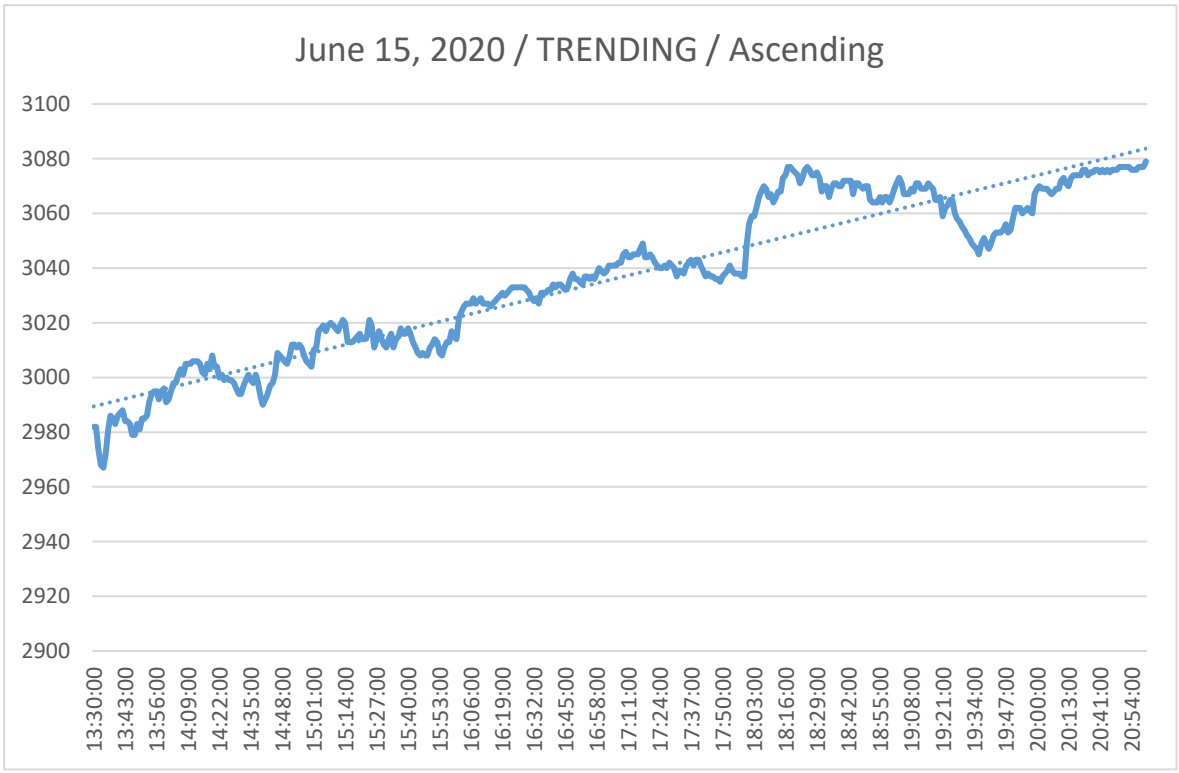
-RANGING / Random Patterns: the chart is “unreadable” (and un-tradable at stage D)(“the market is going nowhere”)



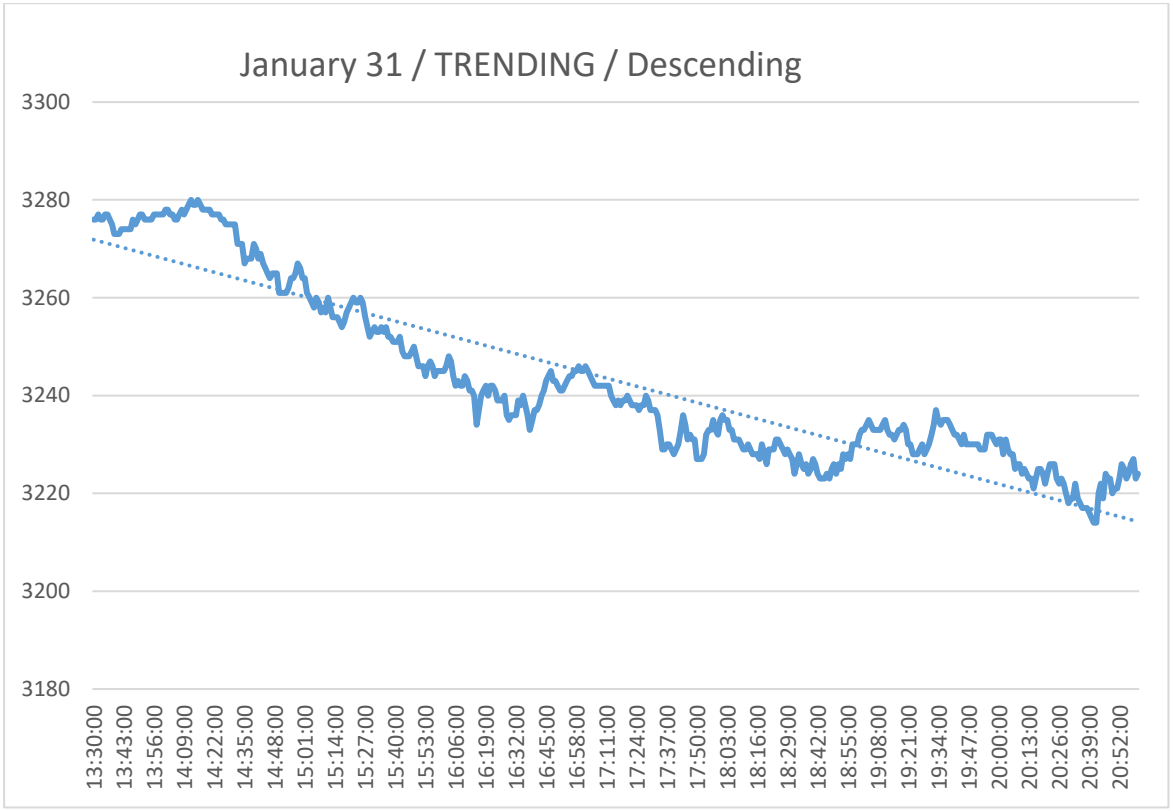
Let us now take cases, from our database in Appendix II:

Cases of TRENDING

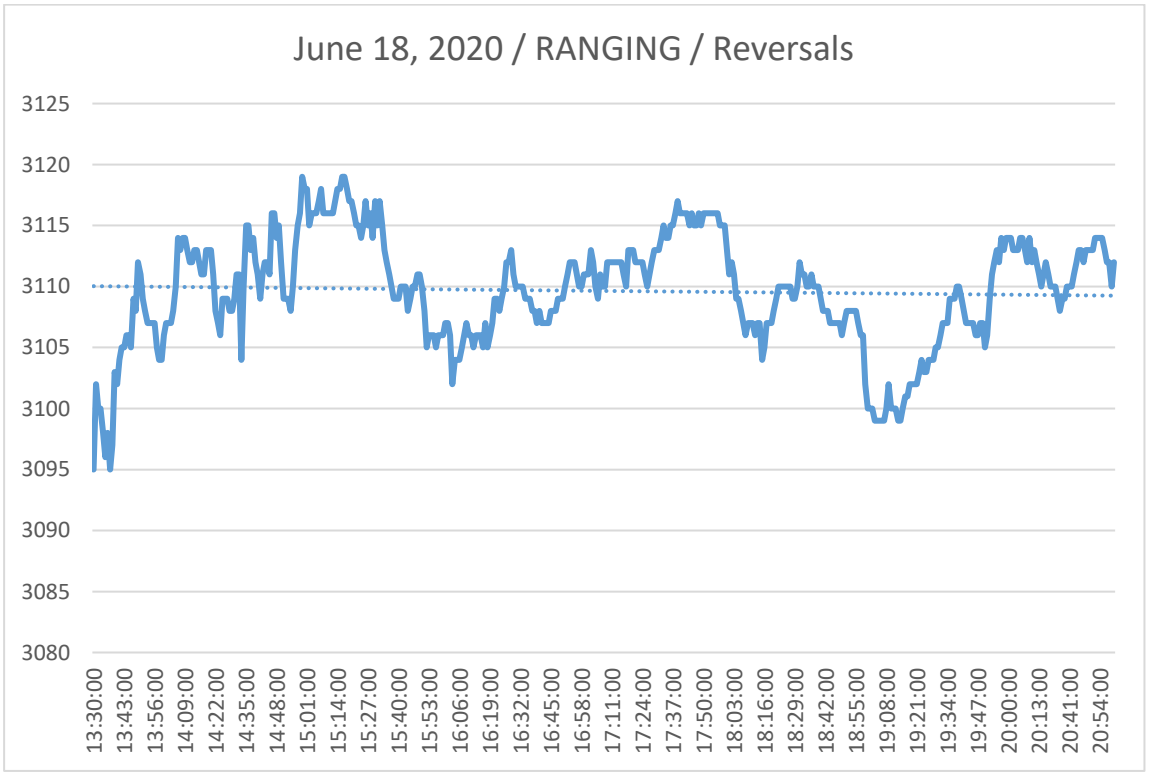
Here is a case of an Ascending Trend



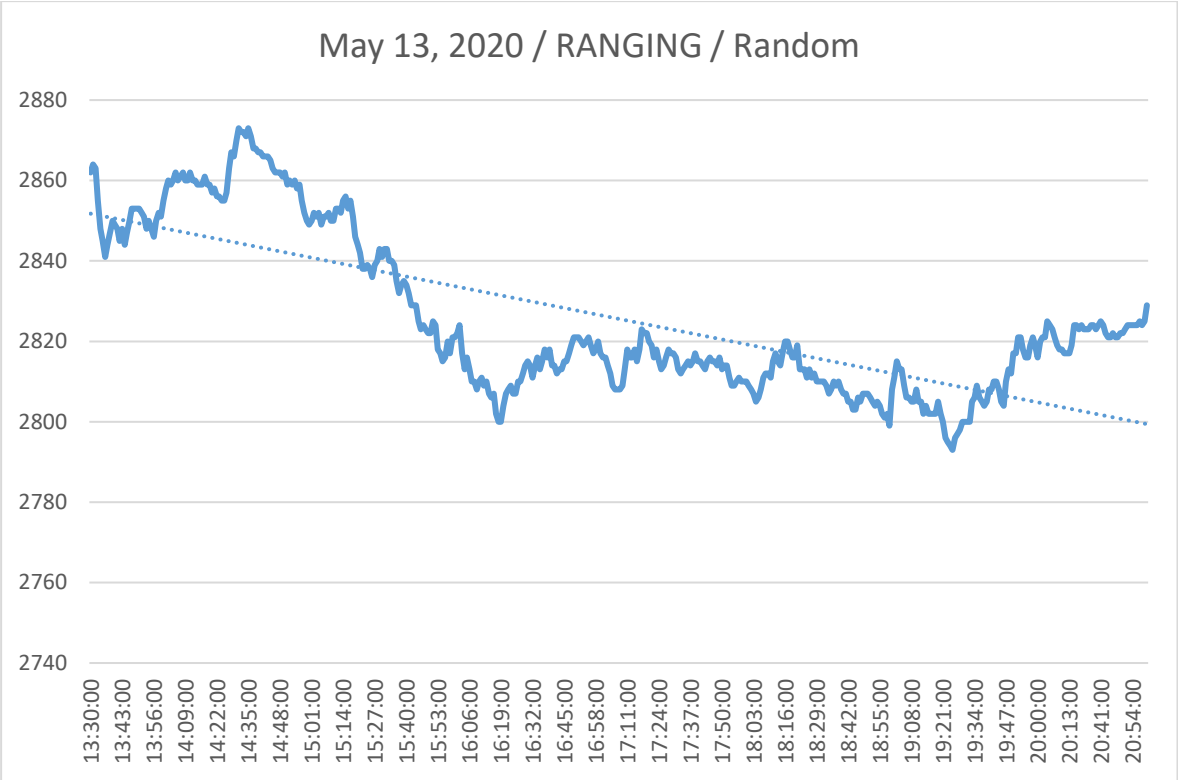
And here is a case of a Descending Trend



Cases of a price RANGING



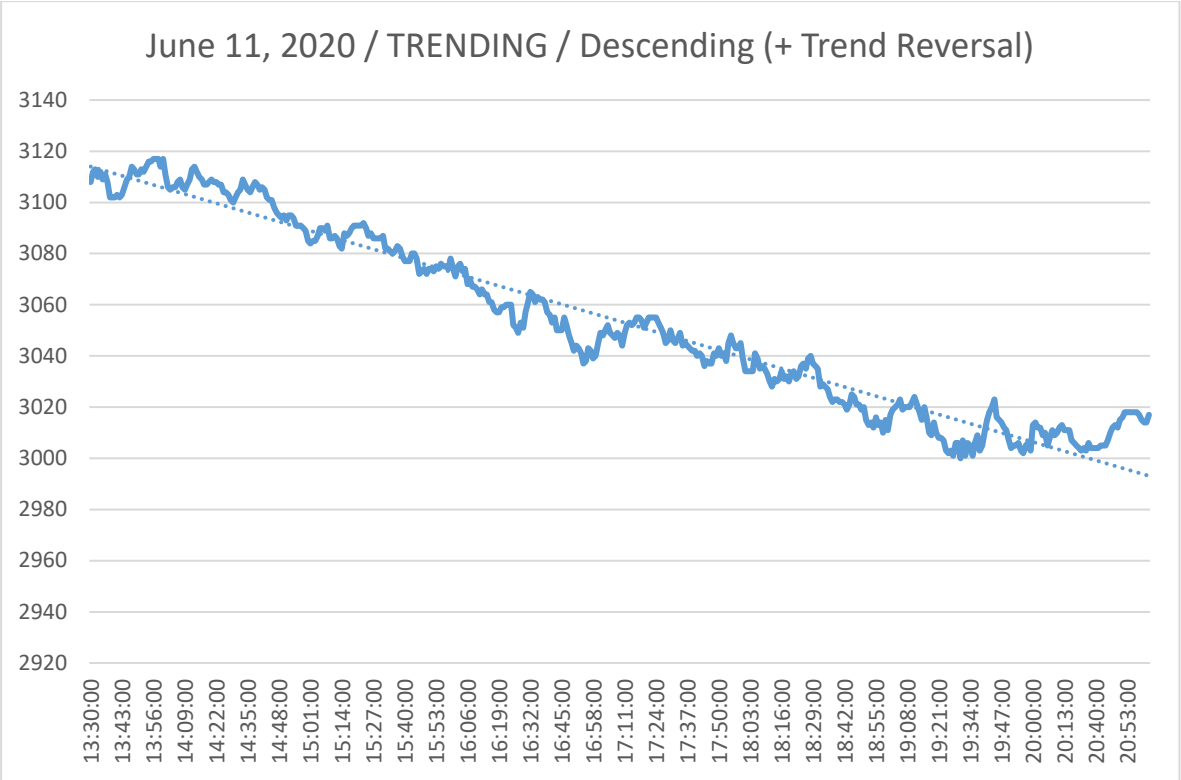
Case of range-bound, random behavior



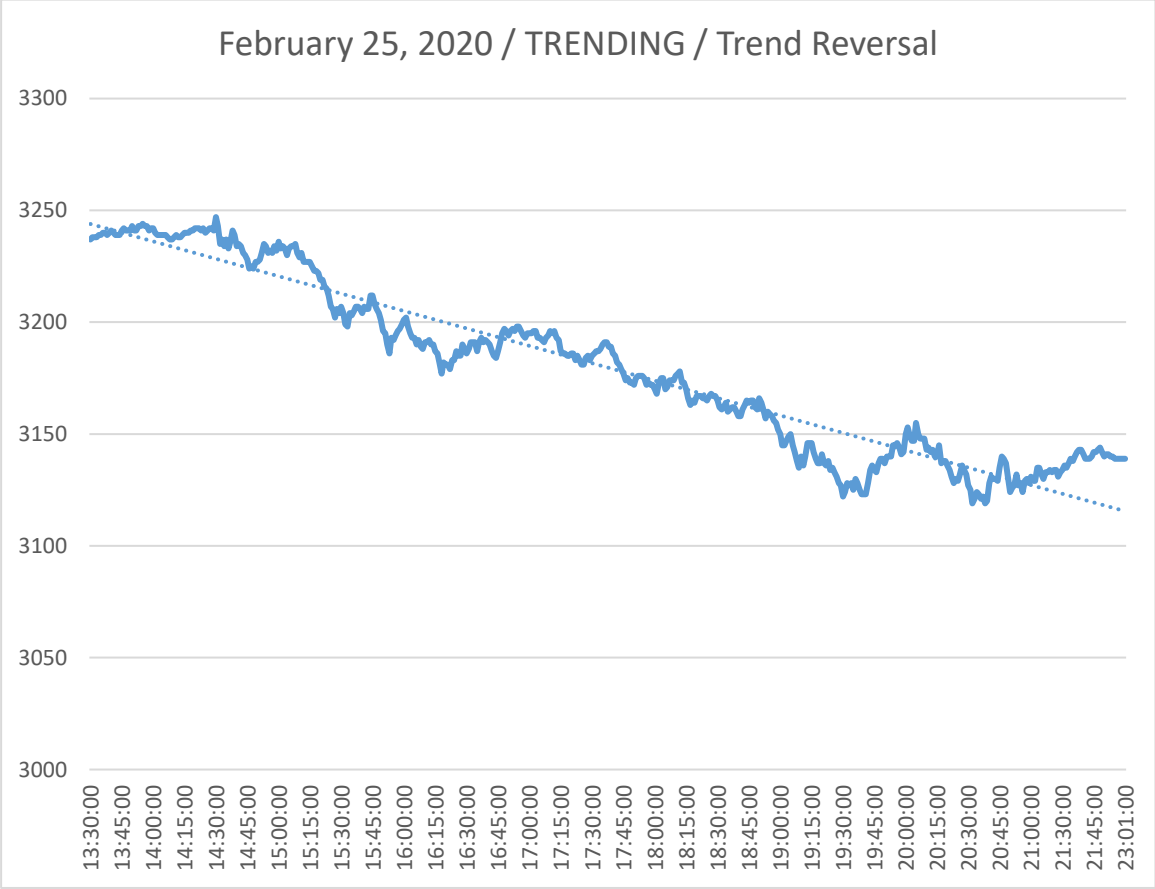
5.5. STAGE D: ACTUAL DECISION-MAKING

There are two complicating factors that make forecasting difficult:

-More often than not we do not encounter pure forms. Worse, pure forms can be misleading. Take the case

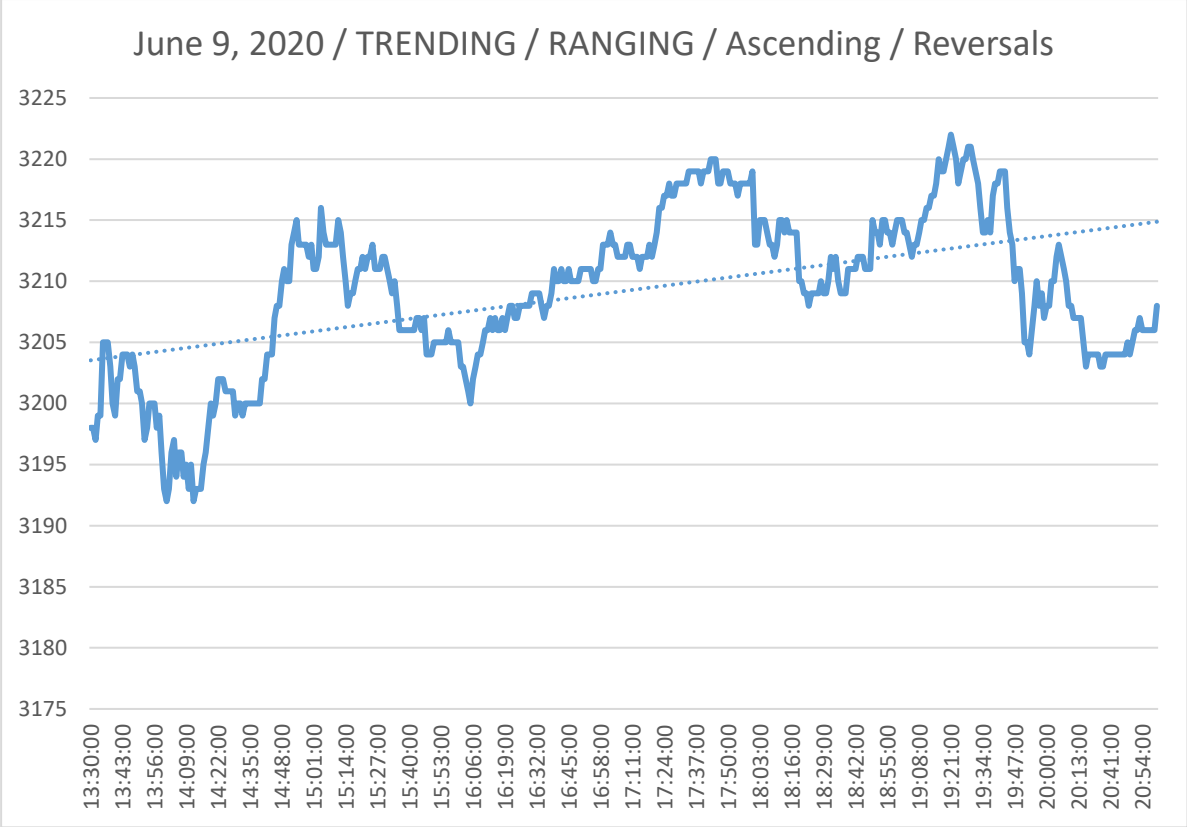


-When it comes to decision-making the trader has to decide on incomplete information



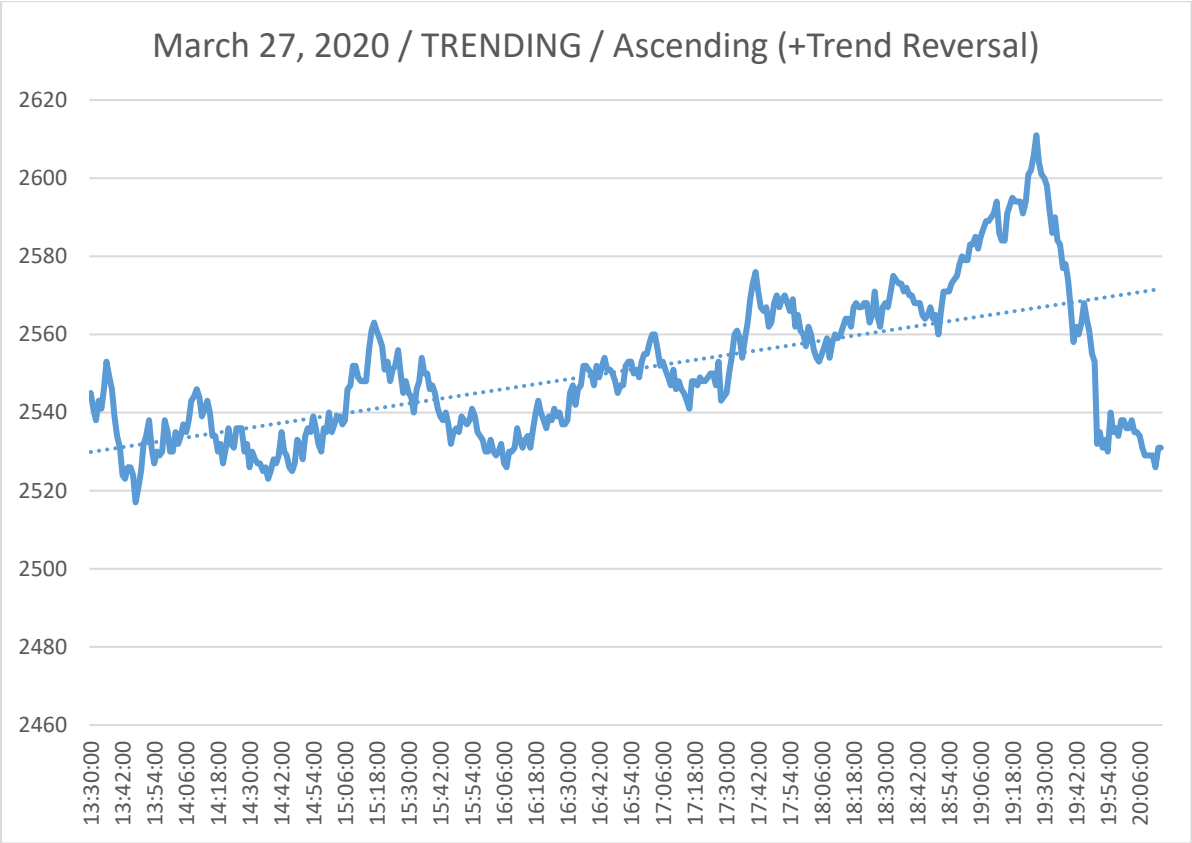
5.5.1. A combination of elementary patterns

Usually there is some kind of combination of the elementary outcomes presented above.

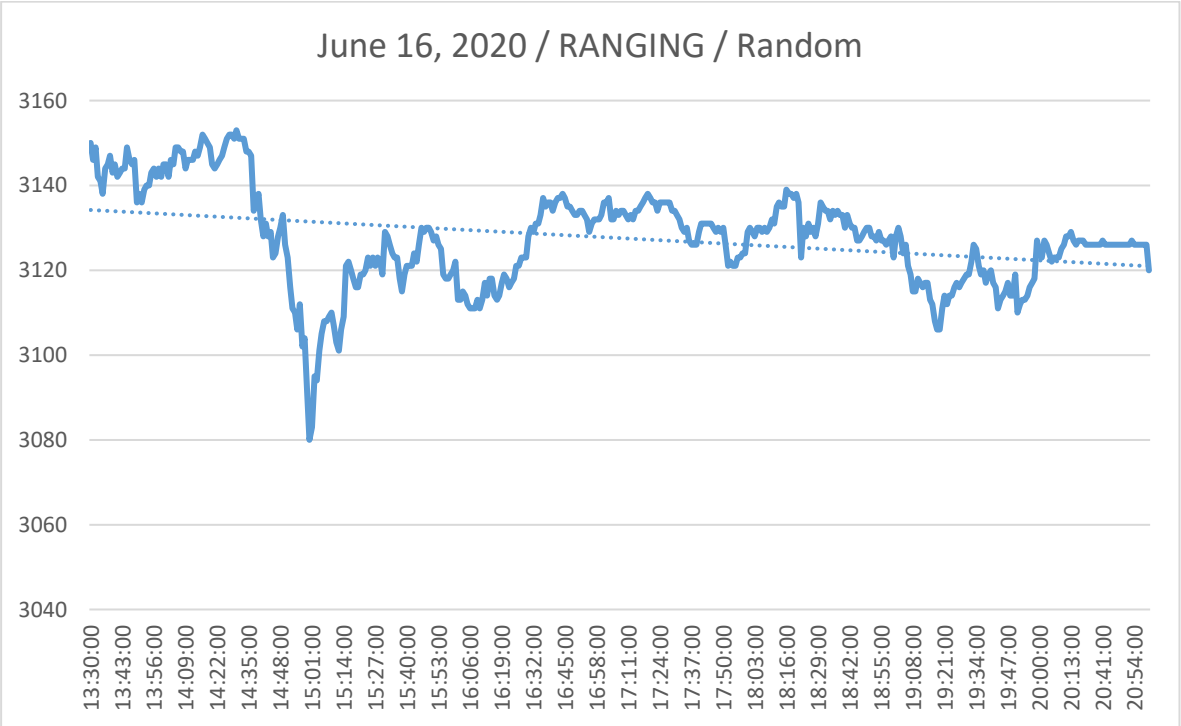


5.5.2. Decision under incomplete information

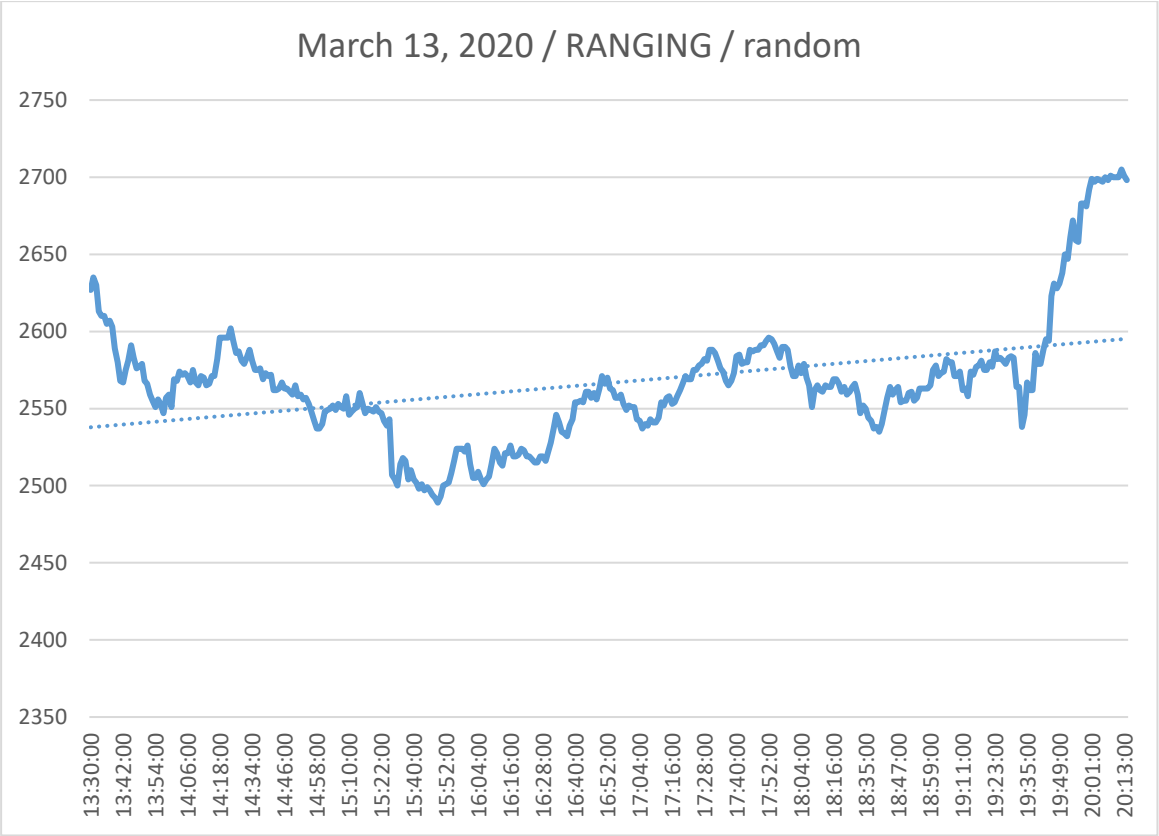
A day-trader must decide within the time frame of the day. He has to decide usually at the second half of the day, when enough data have come in to shape a pattern



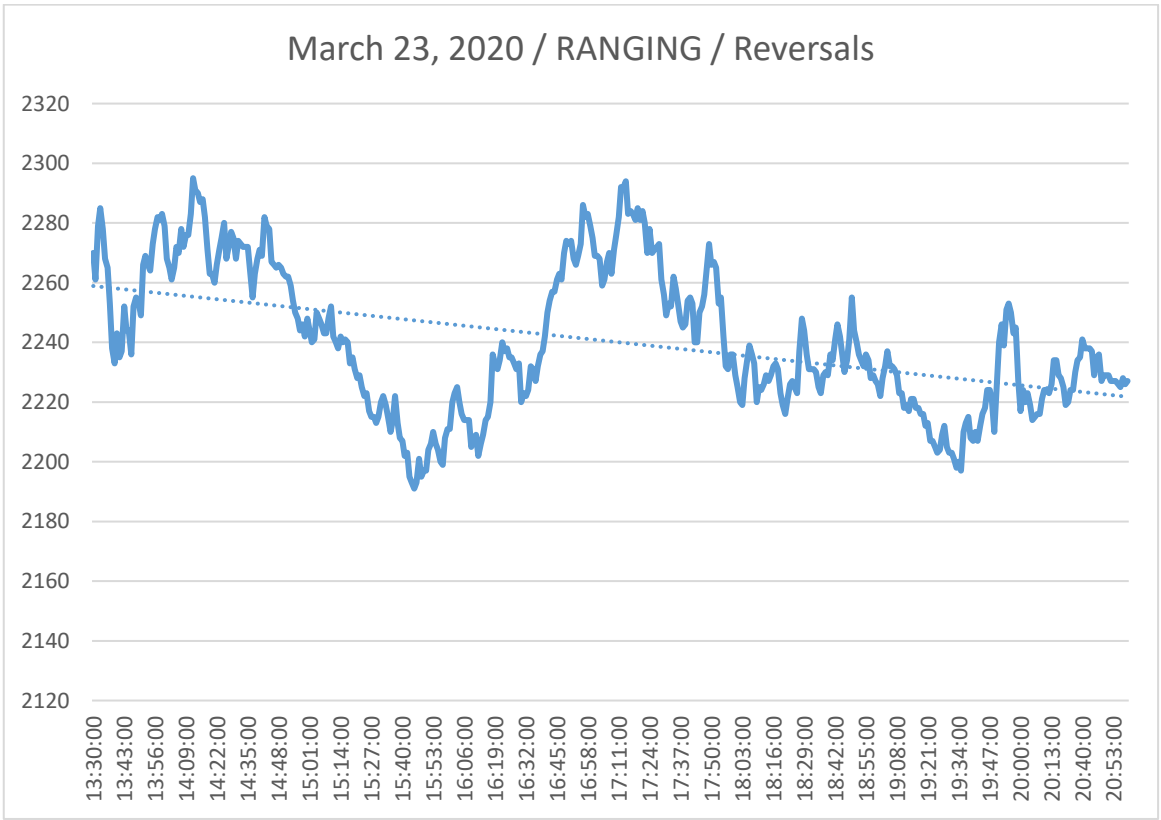
Unreadable (this market is going nowhere)



Untradable



Tradable/ Volatility is Trader's best friend



5.6. Our main empirical result

Trend is more often confirmed than reversed

Out of the 126 trading days of the first half 2020, there were 88 TRENDING days and 38 RANGING days.

Of the 88 TRENDING days, in 57 days the trend was confirmed whereas in 31 days it was reversed.

This is a non trivial result although it needs confirmation.

6. FURTHER RESEARCH

Provide a scientific basis to the so-called "chartist" or "technical" analysis is an underdeveloped area of finance. Time series analysis, even when focusing on stock prices usually stop short of delving into the specificities of trading data, especially intraday data which experience constant growth stemming from the rise of intraday trading activity.

Below we propose a research project, using intraday trading data, that would complete the research carried above.

SPECIFICATIONS FOR A RESEARCH PROJECT IN INTRADAY TRADING

There was a saying in Wall Street that when "everything goes down, the only thing that increases is the correlation between markets". Moreover, under the combined pressure of the forces of globalization and of the rise of Asiatic stock exchanges, correlation between markets has increased to the intraday level between international markets, offering new intraday trading opportunities. By the time NYSE opens (at 14h30 UK time), there have been several hours of trading going on in Hong Kong and Shangai (opening at 3am UK time) as well as in London, Paris and Frankfort (opening around 8am UK time). *Ceteris paribus*, it is frequently observed that the daily % change of closing prices of US index will be situated somewhere (usually at mid-distance) between the "Asiatic" (say Chinese) and European Index closing price. (s. for example attached the daily performance of the Dow , 0,54%, compared to European and Asian markets at close of Friday 7/2/20).

An intraday trading strategy that could yield decent returns would therefore be to construct a simple model with inputs the closing price (at 10am UK time) of Chinese markets, regularly (say, every 30 or 15 minutes) updated with the evolution of the European markets (till their closing at 4 or 5pm UK time, while the US remains open till 9 pm UK time) and output a regularly updated forecast of the US closing price. All other things equal (ie without any significant "local", ie US, event) trading opportunities will arise when the US price diverges intra-session from the forecasted US closing price.

Requirements:

1. Source data from Bloomberg or other source: ideally connect directly to the Bloomberg page (s.below) at 3am and start inputing prices from Asiatic data; then connect to European markets between 7 and 8 am. [Alternatively, simply use historic data available on these markets, available eg at Yahoo finance, imported to Matlab]

2. Produce an average of the Asiatic markets [let us choose just Honk Kong and Shangai for our purpose] and update that as new prices arrive every say 30 minutes

3. Repeat (2) for European markets [let us take the average of FTSE, CAC and DAX]

4. Produce and average of (2) and (3) [this average will include only (2) from 3am to 8am, then (2) and (3) from 8am on]

5. Use (4) to determine the "system model" of the Kalman Filter

6. Use (5) to produce a regularly updated Kalman filter estimate of the expected closing % change of the US index [choose the DOW or the SP500]

INPUT DATA: https://www.bloomberg.com/markets/stocks

Bloomberg Terminal.

NAME	VALUE	NET CHANGE	% CHANGE	1 MONTH	1 YEAR
	29,102.51	-277.26	-0.94%	+0.97%	+15.92%
INDU:IND DOW JONES INDUS. AVG					
	3,327.71	-18.07	-0.54%	+1.91%	+22.89%
SPX:IND S&P 500 INDEX					
	9,520.51	-51.64	-0.54%	+3.72%	+30.45%
CCMP:IND NASDAQ COMPOSITE INDEX					
	13,931.93	-103.02	-0.73%	-0.19%	+13.34%
NYA:IND NYSE COMPOSITE INDEX					
	17,655.49	-102.00	-0.57%	+2.44%	+12.93%
SPTSX:IND S&P/TSX COMPOSITE INDEX					

Europe, Middle East & Africa

NAME	VALUE	NET CHANGE	% CHANGE	1 MONTH	1 YEAR
SX5E:IND Euro Stoxx 50 Pr	3,798.49	-7.03	-0.18%	+0.24%	+21.14%
UKX:IND FTSE 100 INDEX	7,466.70	-38.09	-0.51%	-1.60%	+5.59%
DAX:IND DAX INDEX	13,513.81	-61.01	-0.45%	+0.23%	+23.90%
CAC:IND CAC 40 INDEX	6,029.75	-8.43	-0.14%	-0.12%	+21.53%
IBEX:IND IBEX 35 INDEX	9,811.00	-0.30	0.00%	+2.48%	+10.77%

Asia Pacific

NAME	VALUE	NET CHANGE	% CHANGE	1 MONTH	1 YEAR
NKY:IND NIKKEI 225	23,827.98	-45.61	-0.19%	-0.09%	+17.19%
TPX:IND TOPIX INDEX (TOKYO)	1,732.14	-4.84	-0.28%	-0.17%	+12.52%
HSI:IND HANG SENG INDEX	27,404.27	-89.43	-0.33%	-4.31%	-1.94%
SHSZ300:IND CSI 300 INDEX	3,899.87	+0.09	+0.00%	-6.32%	+20.09%
AS51:IND S&P/ASX 200 INDEX	7,022.58	-26.62	-0.38%	+1.35%	+15.67%
MXAP:IND MSCI AC ASIA PACIFIC	169.69	-0.99	-0.58%	-1.95%	+9.57%

REFERENCES

[Abarbanel, 1996] Abarbanel ,H.D.I. (1996). *Analysis of Observed Chaotic Data*, Springer, New York.

[Aguilar and West, 2000] Aguilar,O. and West,M. (2000). "Bayesian Dynamic Factor Models and Portfolio Allocation", Journal of Business and Economic Statistics, vol.18, pp.338-357.

[Andersen, 2009] Andersen,T. ed. (2009). *Handbook of Financial Time Series*, Springer, Heidelberg, Germany.

[Anderson and Moore, 1979] Anderson,B. and Moore,J. (1979). *Optimal Filtering*, Dover, Mineola, NY.

[Appel, 1981] Appel, G. (1981). Ninety Nine Ways to Make Money in a Depression by Gerald. Appel

[Appel, 1986] Appel,G. (1986). Winning Market Systems: 83 Ways to Beat the Market by Gerald Appel

[Appel, 2005] Appel, G. (2005). *Technical Analysis: Power Tools for Active Investors*

[Appel, 2006] Appel,G., (2006). Opportunity Investing: How to Profit When Stocks Advance, Stocks Decline, Inflation Runs Rampant, Prices Fall

[Appel, 2008a] Appel,G. (2008). Beat the Market: Win with Proven Stock Selection and Market Timing Tools: Win with Proven Stock Selection and... by Gerald Appel

[Appel, G., 2008b] Appel,G. (2008). Beating the Market, 3 Months at a Time: A Proven Investing Plan Everyone Can Use by Gerald Appel and Marvin Appel (17 Jan 2008)

[Aronson, 2007] Aronson, D.R. (2007). *Evidence-Based Technical Analysis*, John Wiley & Sons, Inc., Hoboken, New Jersey.

[Avellaneda, 2010] Avellaneda,M. and Lee,J.H. (2010). "Statistical arbitrage in the US equities market", Quantitative Finance, vol.10(7).

[Bishop, 1995] Bishop, C. (1995). *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, UK.

[Box and Jenkins, 1970] Box,G.E.P. and Jenkins,G.M. (1970). *Time Series Analysis Forecasting and Control*, Holden-Day, San Francisco, CA.

[Briegel and Tresp, 2000] Briegel,Th. and Tresp,V. (2000). "Dynamic Neural Regression Models", Discussion Paper 181 des Insitut für Statistik der Ludwig Maximilians Universität, München.

[Brown, 2012] Brown, Constance M. (2012). *Technical Analysis for the Trading Professional*, 2nd Edition, McGraw-Hill.

[Burgess, 1999] Burgess, A.N. (1999). "A Computational Methodology for Modelling the Dynamics of Statistical Arbitrage", PhD Thesis, London Business School, UK, 1999.

[Chatfield, 2000] Chatfield,C. (2000). *Time Series Forecasting*, Chapman & Hall/CRC, Boca Raton, Florida.

[Chatfield, 2004] Chatfield,C. (2004). *The Analysis of Time Series*, Chapman & Hall/CRC, Boca Raton, Florida.

[Clouse, Gilles, Horne and Cottrell, 1997] Chouse, Gilles, Horne and Cottrell (1997). "Time delay neural networks", IEEE Trans. On Neural Networks, vol.8, N:5, pp1065-1070.

[Danielsson, 2011] Danielsson,J. (2011). "*Financial Risk Forecasting*", Wiley.

[de Freitas et al., 2000] de Freitas,J.F.G., Niranjan,M. and Gee,A.H. (2000). "Hierarchical Bayesian-Kalman Models for Regularisation and ARD in Sequential Learning, Neural Computation", vol.12, pp.933–953.

[Deboeck, 1994] Deboeck, G. editor (1994*). Trading on the Edge, Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets*, John Wiley, New York, NY.

[Dempster, Laird and Rubin, 1977] Dempster,A.P., Laird,N.M. and Rubin,D.B. (1977) "Maximum Likelihood from Incomplete data via the EM algorithm", *J. Royal Statistical Society*, ser.B., Vol.39. pp.1-38.

[Durbin and Koopman, 2001], Durbin, J., Koopman, S.J. (2001). *Time Series Analysis by State-Space Methods*, Oxford University Press, UK.

[Edwards et al., 2013] Edwards,R.D., Magee,J. and Bassetti,W.H.C. (2013). *Technical Analysis of Stock Trends*, 10th Edition, CRC Press, Boca Raton, Florida.

[Elliott et al., 2005] Elliott,R., van der Hoek,J. and Malcolm,W. (2005). "Pairs Trading", Quantitative Finance, vol.5(3), pp.271-276.

[Fama, 1970] Fama,E. (1970). "Efficient Capital Markets: A Review of Theory and Empirical Work", Journal of Finance 25 (2), pp.383–417.

[Faraway and Chatfield, 1998] Faraway,J. and Chatfield,C. (1998). Time Series Forecasting with Neural Networks: A Comparative Study using the Airline Data, *Applied Statistics*, vol. 47., N:2, 231-250.

[Franses and van Dijk, 2003] Franses Ph. and D. van Dijk (2003). *Non-linear Time Series Models in Empirical Finance*, Cambridge University Press, Cambridge, UK.

[Ghahramani, and Hinton, 1996] Ghahramani,Z. and Hinton,G. (1996). "Parameter Estimation for Linear Dynamical Systems", Technical Report CRG-TR-96-2, University of Toronto.

[Goodwin and Sin, 2009] Goodwin,G. and Sin,K. (2009). *Adaptive Filtering Prediction and Control*, Dover, Mineola, NY.

[Gourieroux and Jasiak, 2001] Gourieroux,Ch. and Jasiak,J. (2001). *Financial Econometrics*, Princeton University Press, Princeton, New Jersey.

[Grewal and Andrews, 2014] Grewal,M.S. and Andrews,A.P. (2014). *Kalman Filtering: Theory and Practice with MATLAB*, 4th Edition, Wiley-IEEE Press Publ.

[Hamilton, 1994] Hamilton,J.D. (1994). *Time Series Analysis*, Princeton University Press.

[Harvey and Koopman, 2009] Harvey,A. and Koopman,S. (2009). Unobserved Components Models in Economics and Finance, *IEEE Control Systems Magazine*, pp.71-81.

[Harvey, 2001] Harvey,A. (2001). *Time Series Models*, Essex, UK.

[Haykin, 2001] Haykin,S. (Ed.) (2001). *Kalman Filtering and Neural Networks*, John Wiley, New York, NY.

[Haykin, 2009] Haykin,S. (2009). *Neural Networks and Learning Machines*, Pearson, 3rd ed., New Jersey.

[Holden et al., 1990] Holden,K., Peel,A. and Thompson,J. (1990). *Economic Forecasting*, Cambridge University Press, Cambridge, UK.

[Hornik et al., 1989] Hornik,K., Stinchcombe,M. and White,H. (1989). "Multilayer feedforward networks are universal approximators", Neural Networks, 2:359--366, 1989.

[Kaufman, 1998] Kaufman,P. (1998). *Trading Systems and Methods*, John Wiley, New York, NY.

[Kaufman, 2013] Kaufman,P.J. (2013). *New Trading Systems and Methods*, 5th Edition, John Wiley & Sons, Inc.

[Ke-Lin Du and Swamy, 2013] Ke-Lin Du and Swamy,M.N.S. (2013). *Neural Networks and Statistical Learning*, Springer Science & Business Media.

[Kestner, 2003] Kestner,L. (2003). *Quantitative Trading Strategies*, McGraw Hill, New York, NY.

[Kim and Nelson, 1999] Kim,C.-J. and Nelson,C.R. (1999). *State-Space Models with Regime Switching Classical and Gibbs-Sampling Approaches with Applications*, The MIT Press.

[Kirkpatrick and Dahlquist, 2013] Kirkpatrick,C.D. and Dahlquist,J.R. (2013). *Technical Analysis The Complete Resource for Financial Market Technicians*, Pearson Education, Inc.

[Koop, 2006] Koop,G. (2006). *Analysis of Economic Data*, John Wiley, West Sussex, UK.

[Kovalerchuk and Vityaev, 2001] Kovalerchuk,B. and Vityaev, E. (2001). *Data Mining in Finance*, Kluwer Academic Publishers, Norwell, Massachusetts.

[Lai and Xing, 2008] Lai,T. and Xing,H. (2008). *Statistical Models and Methods for Financial Markets*, Springer, New York, NY.

[Larose, 2006] Larose,D. (2006). *Data Mining Methods and Models*, John Wiley Interscience, Hoboken, NJ.

[Levy, 2004] Levy,G. (2004). *Computational Finance*, Elsevier, Burlington, MA.

[Lin, Horne, Tino and Giles, 1996] Lin,T. Horne,B.G., Tino,P. and Giles,C.L. (1996). "Learning long-term dependencies is not as difficult with NARX recurrent networks", In Advances in Neural Information Processing Systems 8, D.S.Touretzky et al. (Eds). MIT Press, pp.577-602.

[Luo and Unbehauen, 1998] Luo,F.L. and Unbehauen,R. (1998). *Applied Neural Networks For Signal Processing*, Cambridge University Press, Cambridge, UK.

[Martinez et al., 2014] *From an Artificial Neural Network to a Stock Market Day-Trading System: A Case Study on the BM&F BOVESPA*, Conference Paper, UK.

[Malkiel, 1973] Malkiel,B.G. (1973). *A Random Walk Down Wall Street* (6th ed.), W.W.Norton & Company, Inc.

[Maybeck, 1979] Maybeck,P.S. (1979). *Stochastic Models, Estimation and Control*, Vol.1, Academic Press.

[McLachlan and Krishnan, 1997] McLachlan,G. and Krishnan,T. (1997). *The EM Algorithm and Extensions*, John Wiley, New York, NY.

[McNelis, 2005] McNelis,P. (2005). *Neural Networks in Finance: Gaining Predictive Edge in the Market*, Elsevier, Burlington, MA.

[Miazhynskaia et al., 2005] Miazhynskaia,T., Dockner,E., Frühwirth-Schnatter,S. and Dorffner,G. (2005). "Non-linear volatility modelling in classical and Bayesian frameworks with applications to risk management", In: Adaptive Information Systems and Modelling in Economics and Management Science, Hrsg. A.Taudes, Springer Verlag, Wien, pp.73-98.

[Mills, 1991] Mills,T. (1991). *Time Series Techniques for Economists*, Cambridge University Press, Cambridge, UK.

[Mills, 2002] Mills,T. (2002). *The Econometric Modelling of Financial Time Series*, Cambridge University Press, Cambridge, UK.

[Mirikitani, 2010] Mirikitani,D. (2010). "Sequential Recureent Connectionist Algorithms for Time Series Modelling of Nonlinear Dynamical Systems", PhD Thesis, Goldsmiths College, University of London.

[Montana et al., 2009] Montana,G., Triantafyllopoulos,K. and Tsagaris,T. (2009). "Flexible Least Squares for Temporal Data Mining and Statistical Arbitrage", Expert Systems with Applications, vol.36, N:2, pp.2819-2830.

[Montier, 2007] Montier,J. (2007). Behavioural Investing: A Practitioners Guide to Applying Behavioural Finance, John Wiley & Sons Inc.

[Murphy, 1999] Murphy,J. (1999). Technical Analysis of the Financial Markets. Prentice Hall.

[Nabney, 2004] Nabney,I. (2004). *NETLAB: Algorithms for Pattern Recognition*, Springer, London, UK.

[Neal and Hinton, 1998] Neal,R.M. and Hinton,G.E. (1998). A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants, In: M. I. Jordan (editor) *Learning in Graphical Models*, Dordrecht: Kluwer Academic Publ., pp.355-368.

[Neunier and Zimmermann, 1998] Neunier,R. and ZimmermannH.G. (1998). "How to Train Neural Networks", in *Neural Networks: Tricks of the Trade*, LNCS, pp. 373-423.

[Nikolaev and Iba, 2006] Nikolaev,N. and Iba,H. (2006). *Adaptive Learning of Polynomial Networks*, Springer, New York, NY.

[Nison, 2001] Nison,S. (2001). *Japanese Candlestick Charting Techniques*, 2nd Edition, New York Institute of Finance.

[Nørgaard et al., 2000] Nørgaard,M., Ravn,O., Poulsen,N.K. and Hansen,L.K. (2000). *Neural Networks for Modelling and Control of Dynamic Systems*, Springer-Verlag, London.

[Pham and Liu, 1995] Pham,D.T. and Liu,X. (1995). *Neural Networks for Prediction, Identification and Control*, Springer-Verlag, London, UK.

[Piché et al., 2012] Piché,R., Särkkä,S. and Hartikainen,J. (2012). Recursive Outlier-Robust Filtering and Smoothing for Nonlinear Systems Using the Multivariate Student-t Distribution.

In: *Proc. IEEE Int. Workshop on Machine Learning for Signal Processing* (*MLSP*), IEEE Press.

[Pitt and Shephard, 1999] Pitt,M. and Shephard,N. (1999). "Time Varying Covariances: A Factor Stochastic volatility Approach", In: Bayesian Statistics 6, Edited by J.M. Bernardo,

[Pollock, 1999] Pollock,D.S.G. (1999). *Handbook of Time Series Analysis, Signal Processing and Dynamics*, Academic Press.

[Pring, 1993] Pring,M.J. (1993). *Investment Psychology Explained*, John Wiley & Sons, Inc..

[Pring, 2014] Pring,M.J. (2014). *Technical Analysis Explained*, 5th Edition, McGraw Hill Book Company, New York, NY.

[Rachev et al., 2008] Rachev,S., Hsu,J., Bagasheva,B. and Fabozzi,F. (2008). *Bayesian Methods in Finance*, John Wiley, Hoboken, NJ.

[Rasmussen and Williams, 2006] Rasmussen,C.E. and Williams,C.K.I. (2006) *Gaussian Processes for Machine Learning*, The MIT Press, Cambridge, MA.

[Refenes et al., 1996] Refenes,A-P., Abu-Mostafa,Y., Moody,J. and Weigend,A. (ed). (1996). *Neural Networks in Financial Engineering*, World Scientific, Singapore.

[Ruelle, 1989] Ruelle,D. (1989). *Chaotic Evolution and Strange Attractors*, Cambridge University Press, Cambridge, UK.

[Rumelhart et al., 1986] Rumelhart,D.E., Hinton,G.E., and Williams,R.J. (1986). "Learning internal representations by error propagation", In: Rumelhart,D.E. and McClelland,J. L. (Edsd.): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Volume 1: Foundations, MIT Press, Cambridge, MA. pp 318-362.

[Särkkä and Hartikainen, 2013] Särkkä,S. and Hartikainen,J. (2013). Non-Linear Noise Adaptive Kalman Filtering via Variational Bayes, *IEEE International Workshop on Machine Learning For Signal Processing*.

[Särkkä, 2013] Särkkä,S. (2013). *Bayesian Filtering and Smoothing*. Cambridge University Press, Cambridge, UK.

[Shadbolt and Taylor, 2002] Shadbolt,J. and Taylor,J.G. (Eds) (2002). *Neural Networks and the Financial Markets: Predicting, Combining and Portfolio Optimisation*, Springer.

[Shumway and Stoffer, 1982] Shumway,R.H. and Stoffer,D.S. (1982). "An approach to Time Series modeling, smoothing, and forecasting, using the EM algorithm", *Journal of Time Series Analysis*, vol.3, pp.253-269.

[Shumway and Stoffer, 2011] Shumway,R.H. and Stoffer,D.S. (2011). *Time Series Analysis and its Applications*, Springer.

[Solin and Särkkä, 2015] Solin,A. and Särkkä,S. (2015). "State Space Methods for Efficient Inference in Student-t Process Regression" in: Proceedings of AISTATS, vol.38, pp.885-893.

[Stevens, 2002] Stevens,L. (2002). *Essential Technical Analysis*, John Wiley, New York, NY.

[Takens, 1981] Takens,F. (1981). "Detecting Strange Attractors in Turbulence", In: D.A.Rand and L.-S.Young (Eds.): *Dynamical Systems and Turbulence*, Lecture Notes in Mathematics, vol. 898. Springer-Verlag. pp.366–381.

[Tanizaki, 1996] Tanizaki,H. (1996). *Nonlinear Filters: Estimation and Applications*. 2nd Edition. Springer-Verlag. Berlin

[Taylor, 1986] Taylor,S. (1986). *Modelling Financial Time Series*, John Wiley, New York, NY.

[Ting et al., 2007] Ting,J-A., Theodorou,E. and Schaal,S. (2007), "Learning an Outlier-Robust Kalman Filter", in: Proc. Machine Learning: ECML 2007, Springer, pp.748-756.

[Tino et al., 2001] Tino,P., Schittenkopf,Ch., Dorffner,G. (2001). "Financial Volatility Trading using Recurrent Neural Networks", IEEE Trans. On Neural Networks, 12(4), pp. 865-874.

[Tsay, 2005] Tsay,R. (2005). *Analysis of Financial Time Series*, John Wiley, Hoboken, NJ.

[van der Merwe, 2004] van der Merwe,R. (2004)."Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models", PhD Dissertation, Oregon Health & Science University.

[Von Mises, 1964] Von Mises,R. (1964) *Mathematical Theory of Probability and* Statistics, Academic Press.

[Wan, 1993] Wan,E. (1993). "Time Series Prediction Using a Neural Network with Embedded Tapped Delay-Lines," in: *Predicting the Future and Understanding the Past*, A. Weigend, and N. Gershenfeld (eds.), SFI Studies in the Science of Complexity, Proc., 17, Addison-Wesley.

[Wang, 2009] Wang,P. (2009). *Financial Econometrics*, Routledge, New York, NY.

[Weigend et al., 1990] Weigend,A S., Huberman,B.A. and Rumelhart,D.E. (1990). "Predicting the Future: A Connectionist Approach", International Journal of Neural Systems 1, p. 193-209.

[Weigend and Gershenfeld, 1993] Weigend,A. and Gershenfeld,N. (1993). "The Future of Time Series, Learning and Understanding", in: Time Series Prediction, Addison-Wesley.

[Werbos, 1974] Werbos,P.J. (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, PhD thesis, Harvard University.

[Werbos, 1994] Werbos,P.J. (1994). *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*, John Wiley & Sons.

[West and Harrison, 1999] West,M. and Harrison,J. (1999). *Bayesian Forecasting and Dynamical Models*, Springer.

[Wilkinson, 1997] Wilkinson,C. (1997). *Technically Speaking*, Courtesy of Traders Press Inc.

[Zapranis and Refenes, 1999] Zapranis,A. and Refenes,A-P.N. (1999). *Principles of Neural Model Identification, Selection and Adequacy: with Applications to Financial Econometrics*, Springer-Verlag, London.

[Zhang, 2004] Zhang,P. (2004). *Neural Networks in Business Forecasting*, IRM Press, London, UK.

[Zhu, 2003] Zhu,Q.M. (2003). “A Back Propagation Algorithm to Estimate the Parameters of Non-Linear Dynamic Models”, *Applied Mathematical Modelling*, vol. 27, N:3, pp.169-187.

APPENDIX I

Below we propose an intraday trading model inspired from Kalman Filter.

The model is based on intraday data (1 minute to 30 minute) of the SP500 for the day of the 14th of February 2020.

The “system model” in Kalman Filter parlance is provided here by averaging 1 minute, and 5, 10 and 15 minute Moving Averages, to produce a forecast, updated every minute and valid for each 30 minute period.

NB: measurements are supposed to be exact here, so measurement errors are irrelevant here. And the intraday price of the index is supposed to follow a Gaussian with 0 mean, consistent with standard financial theory.

Time	CLOSE 1min	MA5min	MA10min	MA15min	Average 1,5,10,15	KF
14:30:00	3378					
14:31:00	3377					
14:32:00	3378					
14:33:00	3378					
14:34:00	3378	3377,8				
14:35:00	3378	3377,8				
14:36:00	3374	3377,2				
14:37:00	3375	3376,6				
14:38:00	3375	3376,0				
14:39:00	3375	3375,4	3376,6			
14:40:00	3375	3374,8	3376,3			
14:41:00	3375	3375,0	3376,1			
14:42:00	3375	3375,0	3375,8			
14:43:00	3375	3375,0	3375,5			
14:44:00	3374	3374,8	3375,1	3376,0	3375,0	
14:45:00	3376	3375,0	3374,9	3375,9	3375,4	3375,0
14:46:00	3376	3375,2	3375,1	3375,8	3375,5	3375,4
14:47:00	3376	3375,4	3375,2	3375,7	3375,6	3375,5
14:48:00	3376	3375,6	3375,3	3375,5	3375,6	3375,6
14:49:00	3375	3375,8	3375,3	3375,3	3375,4	3375,6
14:50:00	3374	3375,4	3375,2	3375,1	3374,9	3375,4
14:51:00	3373	3374,8	3375,0	3375,0	3374,5	3374,9
14:52:00	3375	3374,6	3375,0	3375,0	3374,9	3374,5
14:53:00	3375	3374,4	3375,0	3375,0	3374,9	3374,9
14:54:00	3374	3374,2	3375,0	3374,9	3374,5	3374,9

14:55:00	3373	3374,0	3374,7	3374,8	3374,1	3374,5
14:56:00	3372	3373,8	3374,3	3374,6	3373,7	3374,1
14:57:00	3373	3373,4	3374,0	3374,5	3373,7	3373,7
14:58:00	3373	3373,0	3373,7	3374,3	3373,5	3373,7
14:59:00	3372	3372,6	3373,4	3374,2	3373,1	3373,5
15:00:00	3374	3372,8	3373,4	3374,1	3373,6	3373,1
15:01:00	3373	3373,0	3373,4	3373,9	3373,3	3373,6
15:02:00	3372	3372,8	3373,1	3373,6	3372,9	3373,3
15:03:00	3374	3373,0	3373,0	3373,5	3373,4	3372,9
15:04:00	3373	3373,2	3372,9	3373,3	3373,1	3373,4
15:05:00	3372	3372,8	3372,8	3373,2	3372,7	3373,1
15:06:00	3372	3372,6	3372,8	3373,1	3372,6	3372,7
15:07:00	3372	3372,6	3372,7	3372,9	3372,6	3372,6
15:08:00	3372	3372,2	3372,6	3372,7	3372,4	3372,6
15:09:00	3373	3372,2	3372,7	3372,7	3372,6	3372,4
15:10:00	3372	3372,2	3372,5	3372,6	3372,3	3372,6
15:11:00	3371	3372,0	3372,3	3372,5	3372,0	3372,3
15:12:00	3370	3371,6	3372,1	3372,3	3371,5	3372,0
15:13:00	3369	3371,0	3371,6	3372,1	3370,9	3371,5
15:14:00	3371	3370,6	3371,4	3372,0	3371,3	3370,9
15:15:00	3372	3370,6	3371,4	3371,9	3371,5	3371,3
15:16:00	3372	3370,8	3371,4	3371,8	3371,5	3371,5
15:17:00	3372	3371,2	3371,4	3371,8	3371,6	3371,5
15:18:00	3373	3372,0	3371,5	3371,7	3372,1	3371,6
15:19:00	3374	3372,6	3371,6	3371,8	3372,5	3372,1
15:20:00	3374	3373,0	3371,8	3371,9	3372,7	3372,5
15:21:00	3375	3373,6	3372,2	3372,1	3373,2	3372,7
15:22:00	3375	3374,2	3372,7	3372,3	3373,6	3373,2
15:23:00	3374	3374,4	3373,2	3372,5	3373,5	3373,6
15:24:00	3374	3374,4	3373,5	3372,5	3373,6	3373,5
15:25:00	3374	3374,4	3373,7	3372,7	3373,7	3373,6
15:26:00	3374	3374,2	3373,9	3372,9	3373,7	3373,7
15:27:00	3375	3374,2	3374,2	3373,2	3374,2	3373,7
15:28:00	3375	3374,4	3374,4	3373,6	3374,4	3374,2
15:29:00	3375	3374,6	3374,5	3373,9	3374,5	3374,4
15:30:00	3375	3374,8	3374,6	3374,1	3374,6	3374,5
15:31:00	3375	3375,0	3374,6	3374,3	3374,7	3374,6
15:32:00	3375	3375,0	3374,6	3374,5	3374,8	3374,7
15:33:00	3375	3375,0	3374,7	3374,6	3374,8	3374,8
15:34:00	3375	3375,0	3374,8	3374,7	3374,9	3374,8
15:35:00	3375	3375,0	3374,9	3374,7	3374,9	3374,9
15:36:00	3376	3375,2	3375,1	3374,8	3375,3	3374,9
15:37:00	3375	3375,2	3375,1	3374,8	3375,0	3375,3
15:38:00	3374	3375,0	3375,0	3374,8	3374,7	3375,0
15:39:00	3375	3375,0	3375,0	3374,9	3375,0	3374,7

15:40:00	3375	3375,0	3375,0	3374,9	3375,0	3375,0
15:41:00	3375	3374,8	3375,0	3375,0	3375,0	3375,0
15:42:00	3375	3374,8	3375,0	3375,0	3375,0	3375,0
15:43:00	3374	3374,8	3374,9	3374,9	3374,7	3375,0
15:44:00	3374	3374,6	3374,8	3374,9	3374,6	3374,7
15:45:00	3374	3374,4	3374,7	3374,8	3374,5	3374,6
15:46:00	3375	3374,4	3374,6	3374,8	3374,7	3374,5
15:47:00	3376	3374,6	3374,7	3374,9	3375,0	3374,7
15:48:00	3376	3375,0	3374,9	3374,9	3375,2	3375,0
15:49:00	3376	3375,4	3375,0	3375,0	3375,4	3375,2
15:50:00	3377	3376,0	3375,2	3375,1	3375,8	3375,4
15:51:00	3377	3376,4	3375,4	3375,2	3376,0	3375,8
15:52:00	3377	3376,6	3375,6	3375,3	3376,1	3376,0
15:53:00	3377	3376,8	3375,9	3375,5	3376,3	3376,1
15:54:00	3376	3376,8	3376,1	3375,6	3376,1	3376,3
15:55:00	3377	3376,8	3376,4	3375,7	3376,5	3376,1
15:56:00	3377	3376,8	3376,6	3375,9	3376,6	3376,5
15:57:00	3377	3376,8	3376,7	3376,0	3376,6	3376,6
15:58:00	3377	3376,8	3376,8	3376,2	3376,7	3376,6
15:59:00	3377	3377,0	3376,9	3376,4	3376,8	3376,7
16:00:00	3377	3377,0	3376,9	3376,6	3376,9	3376,8
16:01:00	3377	3377,0	3376,9	3376,7	3376,9	3376,9
16:02:00	3377	3377,0	3376,9	3376,8	3376,9	3376,9
16:03:00	3377	3377,0	3376,9	3376,9	3376,9	3376,9
16:04:00	3377	3377,0	3377,0	3376,9	3377,0	3376,9
16:05:00	3377	3377,0	3377,0	3376,9	3377,0	3377,0
16:06:00	3377	3377,0	3377,0	3376,9	3377,0	3377,0
16:07:00	3377	3377,0	3377,0	3376,9	3377,0	3377,0
16:08:00	3376	3376,8	3376,9	3376,9	3376,6	3377,0
16:09:00	3376	3376,6	3376,8	3376,9	3376,6	3376,6
16:10:00	3376	3376,4	3376,7	3376,8	3376,5	3376,6
16:11:00	3377	3376,4	3376,7	3376,8	3376,7	3376,5
16:12:00	3377	3376,4	3376,7	3376,8	3376,7	3376,7
16:13:00	3378	3376,8	3376,8	3376,9	3377,1	3376,7
16:14:00	3378	3377,2	3376,9	3376,9	3377,3	3377,1
16:15:00	3378	3377,6	3377,0	3377,0	3377,4	3377,3
16:16:00	3378	3377,8	3377,1	3377,1	3377,5	3377,4
16:17:00	3378	3378,0	3377,2	3377,1	3377,6	3377,5
16:18:00	3379	3378,2	3377,5	3377,3	3378,0	3377,6
16:19:00	3379	3378,4	3377,8	3377,4	3378,2	3378,0
16:20:00	3378	3378,4	3378,0	3377,5	3378,0	3378,2
16:21:00	3378	3378,4	3378,1	3377,5	3378,0	3378,0
16:22:00	3378	3378,4	3378,2	3377,6	3378,1	3378,0
16:23:00	3378	3378,2	3378,2	3377,7	3378,0	3378,1
16:24:00	3378	3378,0	3378,2	3377,9	3378,0	3378,0

16:25:00	3377	3377,8	3378,1	3377,9	3377,7	3378,0
16:26:00	3377	3377,6	3378,0	3377,9	3377,6	3377,7
16:27:00	3377	3377,4	3377,9	3377,9	3377,6	3377,6
16:28:00	3377	3377,2	3377,7	3377,9	3377,4	3377,6
16:29:00	3377	3377,0	3377,5	3377,8	3377,3	3377,4
16:30:00	3376	3376,8	3377,3	3377,7	3376,9	3377,3
16:31:00	3377	3376,8	3377,2	3377,6	3377,2	3376,9
16:32:00	3377	3376,8	3377,1	3377,5	3377,1	3377,2
16:33:00	3377	3376,8	3377,0	3377,4	3377,1	3377,1
16:34:00	3377	3376,8	3376,9	3377,3	3377,0	3377,1
16:35:00	3377	3377,0	3376,9	3377,2	3377,0	3377,0
16:36:00	3377	3377,0	3376,9	3377,1	3377,0	3377,0
16:37:00	3377	3377,0	3376,9	3377,1	3377,0	3377,0
16:38:00	3377	3377,0	3376,9	3377,0	3377,0	3377,0
16:39:00	3377	3377,0	3376,9	3376,9	3377,0	3377,0
16:40:00	3377	3377,0	3377,0	3376,9	3377,0	3377,0
16:41:00	3378	3377,2	3377,1	3377,0	3377,3	3377,0
16:42:00	3378	3377,4	3377,2	3377,1	3377,4	3377,3
16:43:00	3377	3377,4	3377,2	3377,1	3377,2	3377,4
16:44:00	3377	3377,4	3377,2	3377,1	3377,2	3377,2
16:45:00	3377	3377,4	3377,2	3377,1	3377,2	3377,2
16:46:00	3377	3377,2	3377,2	3377,1	3377,1	3377,2
16:47:00	3377	3377,0	3377,2	3377,1	3377,1	3377,1
16:48:00	3377	3377,0	3377,2	3377,1	3377,1	3377,1
16:49:00	3375	3376,6	3377,0	3377,0	3376,4	3377,1
16:50:00	3375	3376,2	3376,8	3376,9	3376,2	3376,4
16:51:00	3375	3375,8	3376,5	3376,7	3376,0	3376,2
16:52:00	3375	3375,4	3376,2	3376,6	3375,8	3376,0
16:53:00	3375	3375,0	3376,0	3376,5	3375,6	3375,8
16:54:00	3375	3375,0	3375,8	3376,3	3375,5	3375,6
16:55:00	3375	3375,0	3375,6	3376,2	3375,5	3375,5
16:56:00	3376	3375,2	3375,5	3376,1	3375,7	3375,5
16:57:00	3376	3375,4	3375,4	3375,9	3375,7	3375,7
16:58:00	3376	3375,6	3375,3	3375,9	3375,7	3375,7
16:59:00	3375	3375,6	3375,3	3375,7	3375,4	3375,7
17:00:00	3375	3375,6	3375,3	3375,6	3375,4	3375,4
17:01:00	3376	3375,6	3375,4	3375,5	3375,6	3375,4
17:02:00	3376	3375,6	3375,5	3375,5	3375,6	3375,6
17:03:00	3375	3375,4	3375,5	3375,3	3375,3	3375,6
17:04:00	3374	3375,2	3375,4	3375,3	3375,0	3375,3
17:05:00	3375	3375,2	3375,4	3375,3	3375,2	3375,0
17:06:00	3374	3374,8	3375,2	3375,2	3374,8	3375,2
17:07:00	3373	3374,2	3374,9	3375,1	3374,3	3374,8
17:08:00	3373	3373,8	3374,6	3374,9	3374,1	3374,3
17:09:00	3373	3373,6	3374,4	3374,8	3374,0	3374,1

17:10:00	3374	3373,4	3374,3	3374,7	3374,1	3374,0
17:11:00	3373	3373,2	3374,0	3374,5	3373,7	3374,1
17:12:00	3373	3373,2	3373,7	3374,3	3373,6	3373,7
17:13:00	3373	3373,2	3373,5	3374,1	3373,5	3373,6
17:14:00	3373	3373,2	3373,4	3374,0	3373,4	3373,5
17:15:00	3373	3373,0	3373,2	3373,9	3373,3	3373,4
17:16:00	3373	3373,0	3373,1	3373,7	3373,2	3373,3
17:17:00	3373	3373,0	3373,1	3373,5	3373,1	3373,2
17:18:00	3373	3373,0	3373,1	3373,3	3373,1	3373,1
17:19:00	3373	3373,0	3373,1	3373,3	3373,1	3373,1
17:20:00	3372	3372,8	3372,9	3373,1	3372,7	3373,1
17:21:00	3371	3372,4	3372,7	3372,9	3372,2	3372,7
17:22:00	3370	3371,8	3372,4	3372,7	3371,7	3372,2
17:23:00	3371	3371,4	3372,2	3372,5	3371,8	3371,7
17:24:00	3371	3371,0	3372,0	3372,4	3371,6	3371,8
17:25:00	3371	3370,8	3371,8	3372,2	3371,5	3371,6
17:26:00	3371	3370,8	3371,6	3372,1	3371,4	3371,5
17:27:00	3372	3371,2	3371,5	3372,0	3371,7	3371,4
17:28:00	3371	3371,2	3371,3	3371,9	3371,3	3371,7
17:29:00	3372	3371,4	3371,2	3371,8	3371,6	3371,3
17:30:00	3372	3371,6	3371,2	3371,7	3371,6	3371,6
17:31:00	3372	3371,8	3371,3	3371,7	3371,7	3371,6
17:32:00	3372	3371,8	3371,5	3371,6	3371,7	3371,7
17:33:00	3371	3371,8	3371,5	3371,5	3371,4	3371,7
17:34:00	3372	3371,8	3371,6	3371,4	3371,7	3371,4
17:35:00	3372	3371,8	3371,7	3371,4	3371,7	3371,7
17:36:00	3372	3371,8	3371,8	3371,5	3371,8	3371,7
17:37:00	3372	3371,8	3371,8	3371,6	3371,8	3371,8
17:38:00	3372	3372,0	3371,9	3371,7	3371,9	3371,8
17:39:00	3373	3372,2	3372,0	3371,8	3372,3	3371,9
17:40:00	3373	3372,4	3372,1	3371,9	3372,4	3372,3
17:41:00	3373	3372,6	3372,2	3372,1	3372,5	3372,4
17:42:00	3372	3372,6	3372,2	3372,1	3372,2	3372,5
17:43:00	3372	3372,6	3372,3	3372,1	3372,3	3372,2
17:44:00	3372	3372,4	3372,3	3372,1	3372,2	3372,3
17:45:00	3371	3372,0	3372,2	3372,1	3371,8	3372,2
17:46:00	3372	3371,8	3372,2	3372,1	3372,0	3371,8
17:47:00	3371	3371,6	3372,1	3372,0	3371,7	3372,0
17:48:00	3370	3371,2	3371,9	3371,9	3371,3	3371,7
17:49:00	3371	3371,0	3371,7	3371,9	3371,4	3371,3
17:50:00	3371	3371,0	3371,5	3371,8	3371,3	3371,4
17:51:00	3370	3370,6	3371,2	3371,7	3370,9	3371,3
17:52:00	3370	3370,4	3371,0	3371,5	3370,7	3370,9
17:53:00	3370	3370,4	3370,8	3371,4	3370,7	3370,7
17:54:00	3370	3370,2	3370,6	3371,2	3370,5	3370,7

17:55:00	3369	3369,8	3370,4	3370,9	3370,0	3370,5
17:56:00	3369	3369,6	3370,1	3370,7	3369,8	3370,0
17:57:00	3369	3369,4	3369,9	3370,5	3369,7	3369,8
17:58:00	3370	3369,4	3369,9	3370,3	3369,9	3369,7
17:59:00	3370	3369,4	3369,8	3370,2	3369,9	3369,9
18:00:00	3370	3369,6	3369,7	3370,1	3369,9	3369,9
18:01:00	3370	3369,8	3369,7	3370,0	3369,9	3369,9
18:02:00	3371	3370,2	3369,8	3370,0	3370,3	3369,9
18:03:00	3370	3370,2	3369,8	3370,0	3370,0	3370,3
18:04:00	3371	3370,4	3369,9	3370,0	3370,3	3370,0
18:05:00	3372	3370,8	3370,2	3370,1	3370,8	3370,3
18:06:00	3371	3371,0	3370,4	3370,1	3370,6	3370,8
18:07:00	3371	3371,0	3370,6	3370,2	3370,7	3370,6
18:08:00	3369	3370,8	3370,5	3370,1	3370,1	3370,7
18:09:00	3368	3370,2	3370,3	3370,0	3369,6	3370,1
18:10:00	3368	3369,4	3370,1	3369,9	3369,4	3369,6
18:11:00	3370	3369,2	3370,1	3370,0	3369,8	3369,4
18:12:00	3370	3369,0	3370,0	3370,1	3369,8	3369,8
18:13:00	3370	3369,2	3370,0	3370,1	3369,8	3369,8
18:14:00	3370	3369,6	3369,9	3370,1	3369,9	3369,8
18:15:00	3370	3370,0	3369,7	3370,1	3369,9	3369,9
18:16:00	3368	3369,6	3369,4	3369,9	3369,2	3369,9
18:17:00	3368	3369,2	3369,1	3369,7	3369,0	3369,2
18:18:00	3368	3368,8	3369,0	3369,6	3368,9	3369,0
18:19:00	3368	3368,4	3369,0	3369,4	3368,7	3368,9
18:20:00	3368	3368,0	3369,0	3369,1	3368,5	3368,7
18:21:00	3368	3368,0	3368,8	3368,9	3368,4	3368,5
18:22:00	3368	3368,0	3368,6	3368,7	3368,3	3368,4
18:23:00	3369	3368,2	3368,5	3368,7	3368,6	3368,3
18:24:00	3369	3368,4	3368,4	3368,8	3368,7	3368,6
18:25:00	3369	3368,6	3368,3	3368,9	3368,7	3368,7
18:26:00	3369	3368,8	3368,4	3368,8	3368,8	3368,7
18:27:00	3370	3369,2	3368,6	3368,8	3369,2	3368,8
18:28:00	3370	3369,4	3368,8	3368,8	3369,3	3369,2
18:29:00	3369	3369,4	3368,9	3368,7	3369,0	3369,3
18:30:00	3369	3369,4	3369,0	3368,7	3369,0	3369,0
18:31:00	3369	3369,4	3369,1	3368,7	3369,1	3369,0
18:32:00	3369	3369,2	3369,2	3368,8	3369,1	3369,1
18:33:00	3370	3369,2	3369,3	3368,9	3369,4	3369,1
18:34:00	3370	3369,4	3369,4	3369,1	3369,5	3369,4
18:35:00	3371	3369,8	3369,6	3369,3	3369,9	3369,5
18:36:00	3370	3370,0	3369,7	3369,4	3369,8	3369,9
18:37:00	3371	3370,4	3369,8	3369,6	3370,2	3369,8
18:38:00	3371	3370,6	3369,9	3369,7	3370,3	3370,2
18:39:00	3370	3370,6	3370,0	3369,8	3370,1	3370,3

18:40:00	3370	3370,4	3370,1	3369,9	3370,1	3370,1
18:41:00	3370	3370,4	3370,2	3369,9	3370,1	3370,1
18:42:00	3370	3370,2	3370,3	3369,9	3370,1	3370,1
18:43:00	3370	3370,0	3370,3	3369,9	3370,1	3370,1
18:44:00	3371	3370,2	3370,4	3370,1	3370,4	3370,1
18:45:00	3370	3370,2	3370,3	3370,1	3370,2	3370,4
18:46:00	3370	3370,2	3370,3	3370,2	3370,2	3370,2
18:47:00	3370	3370,2	3370,2	3370,3	3370,2	3370,2
18:48:00	3370	3370,2	3370,1	3370,3	3370,1	3370,2
18:49:00	3369	3369,8	3370,0	3370,2	3369,8	3370,1
18:50:00	3370	3369,8	3370,0	3370,1	3370,0	3369,8
18:51:00	3369	3369,6	3369,9	3370,1	3369,6	3370,0
18:52:00	3370	3369,6	3369,9	3370,0	3369,9	3369,6
18:53:00	3369	3369,4	3369,8	3369,9	3369,5	3369,9
18:54:00	3369	3369,4	3369,6	3369,8	3369,5	3369,5
18:55:00	3369	3369,2	3369,5	3369,7	3369,4	3369,5
18:56:00	3368	3369,0	3369,3	3369,6	3369,0	3369,4
18:57:00	3368	3368,6	3369,1	3369,5	3368,8	3369,0
18:58:00	3368	3368,4	3368,9	3369,3	3368,7	3368,8
18:59:00	3368	3368,2	3368,8	3369,1	3368,5	3368,7
19:00:00	3370	3368,4	3368,8	3369,1	3369,1	3368,5
19:01:00	3370	3368,8	3368,9	3369,1	3369,2	3369,1
19:02:00	3370	3369,2	3368,9	3369,1	3369,3	3369,2
19:03:00	3370	3369,6	3369,0	3369,1	3369,4	3369,3
19:04:00	3370	3370,0	3369,1	3369,2	3369,6	3369,4
19:05:00	3370	3370,0	3369,2	3369,2	3369,6	3369,6
19:06:00	3370	3370,0	3369,4	3369,3	3369,7	3369,6
19:07:00	3370	3370,0	3369,6	3369,3	3369,7	3369,7
19:08:00	3371	3370,2	3369,9	3369,4	3370,1	3369,7
19:09:00	3371	3370,4	3370,2	3369,5	3370,3	3370,1
19:10:00	3371	3370,6	3370,3	3369,7	3370,4	3370,3
19:11:00	3371	3370,8	3370,4	3369,9	3370,5	3370,4
19:12:00	3370	3370,8	3370,4	3370,0	3370,3	3370,5
19:13:00	3370	3370,6	3370,4	3370,1	3370,3	3370,3
19:14:00	3369	3370,2	3370,3	3370,2	3369,9	3370,3
19:15:00	3368	3369,6	3370,1	3370,1	3369,4	3369,9
19:16:00	3365	3368,4	3369,6	3369,7	3368,2	3369,4
19:17:00	3366	3367,6	3369,2	3369,5	3368,1	3368,2
19:18:00	3366	3366,8	3368,7	3369,2	3367,7	3368,1
19:19:00	3366	3366,2	3368,2	3368,9	3367,3	3367,7
19:20:00	3367	3366,0	3367,8	3368,7	3367,4	3367,3
19:21:00	3367	3366,4	3367,4	3368,5	3367,3	3367,4
19:22:00	3367	3366,6	3367,1	3368,3	3367,3	3367,3
19:23:00	3369	3367,2	3367,0	3368,2	3367,9	3367,3
19:24:00	3369	3367,8	3367,0	3368,1	3368,0	3367,9

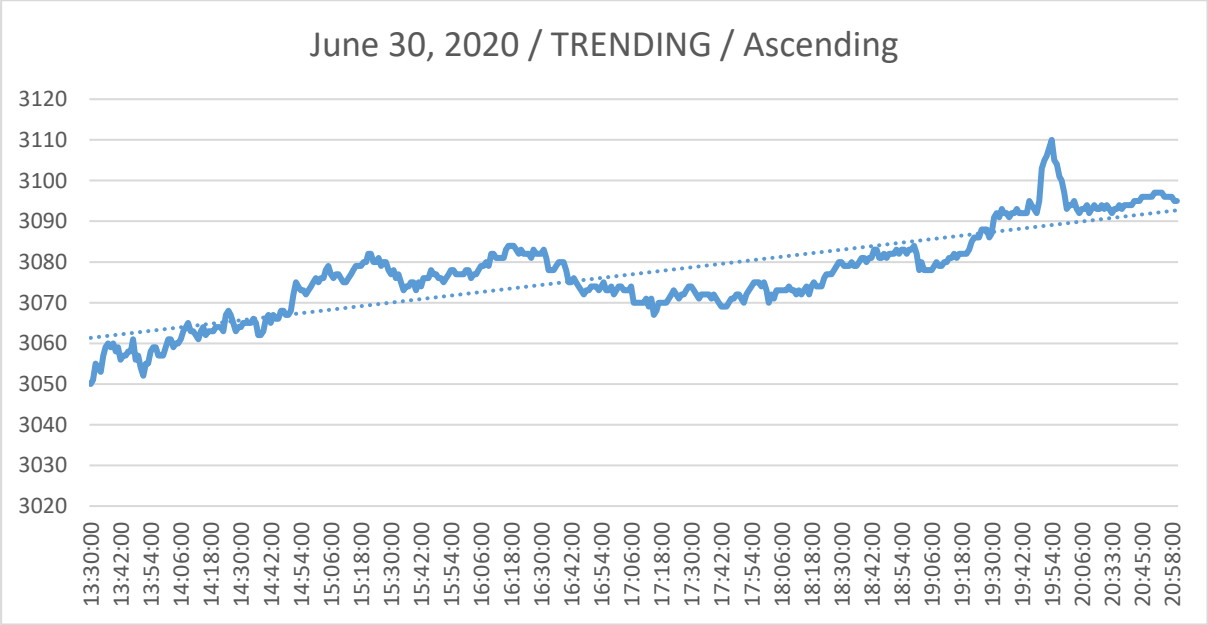
19:25:00	3370	3368,4	3367,2	3368,0	3368,4	3368,0
19:26:00	3371	3369,2	3367,8	3368,0	3369,0	3368,4
19:27:00	3371	3370,0	3368,3	3368,1	3369,3	3369,0
19:28:00	3371	3370,4	3368,8	3368,1	3369,6	3369,3
19:29:00	3371	3370,8	3369,3	3368,3	3369,8	3369,6
19:30:00	3371	3371,0	3369,7	3368,5	3370,0	3369,8
19:31:00	3370	3370,8	3370,0	3368,8	3369,9	3370,0
19:32:00	3370	3370,6	3370,3	3369,1	3370,0	3369,9
19:33:00	3372	3370,8	3370,6	3369,5	3370,7	3370,0
19:34:00	3372	3371,0	3370,9	3369,9	3370,9	3370,7
19:35:00	3373	3371,4	3371,2	3370,3	3371,5	3370,9
19:36:00	3373	3372,0	3371,4	3370,7	3371,8	3371,5
19:37:00	3374	3372,8	3371,7	3371,1	3372,4	3371,8
19:38:00	3374	3373,2	3372,0	3371,5	3372,7	3372,4
19:39:00	3374	3373,6	3372,3	3371,8	3372,9	3372,7
19:40:00	3373	3373,6	3372,5	3372,0	3372,8	3372,9
19:41:00	3372	3373,4	3372,7	3372,1	3372,5	3372,8
19:42:00	3372	3373,0	3372,9	3372,1	3372,5	3372,5
19:43:00	3372	3372,6	3372,9	3372,2	3372,4	3372,5
19:44:00	3372	3372,2	3372,9	3372,3	3372,3	3372,4
19:45:00	3372	3372,0	3372,8	3372,3	3372,3	3372,3
19:46:00	3371	3371,8	3372,6	3372,4	3372,0	3372,3
19:47:00	3371	3371,6	3372,3	3372,5	3371,8	3372,0
19:48:00	3372	3371,6	3372,1	3372,5	3372,0	3371,8
19:49:00	3371	3371,4	3371,8	3372,4	3371,7	3372,0
19:50:00	3371	3371,2	3371,6	3372,3	3371,5	3371,7
19:51:00	3371	3371,2	3371,5	3372,1	3371,5	3371,5
19:52:00	3371	3371,2	3371,4	3371,9	3371,4	3371,5
19:53:00	3371	3371,0	3371,3	3371,7	3371,3	3371,4
19:54:00	3370	3370,8	3371,1	3371,5	3370,8	3371,3
19:55:00	3370	3370,6	3370,9	3371,3	3370,7	3370,8
19:56:00	3371	3370,6	3370,9	3371,2	3370,9	3370,7
19:57:00	3371	3370,6	3370,9	3371,1	3370,9	3370,9
19:58:00	3371	3370,6	3370,8	3371,1	3370,9	3370,9
19:59:00	3371	3370,8	3370,8	3371,0	3370,9	3370,9
20:00:00	3370	3370,8	3370,7	3370,9	3370,6	3370,9
20:01:00	3371	3370,8	3370,7	3370,9	3370,8	3370,6
20:02:00	3371	3370,8	3370,7	3370,9	3370,8	3370,8
20:03:00	3371	3370,8	3370,7	3370,8	3370,8	3370,8
20:04:00	3370	3370,6	3370,7	3370,7	3370,5	3370,8
20:05:00	3370	3370,6	3370,7	3370,7	3370,5	3370,5
20:06:00	3370	3370,4	3370,6	3370,6	3370,4	3370,5
20:07:00	3371	3370,4	3370,6	3370,6	3370,7	3370,4
20:08:00	3371	3370,4	3370,6	3370,6	3370,7	3370,7
20:09:00	3371	3370,6	3370,6	3370,7	3370,7	3370,7

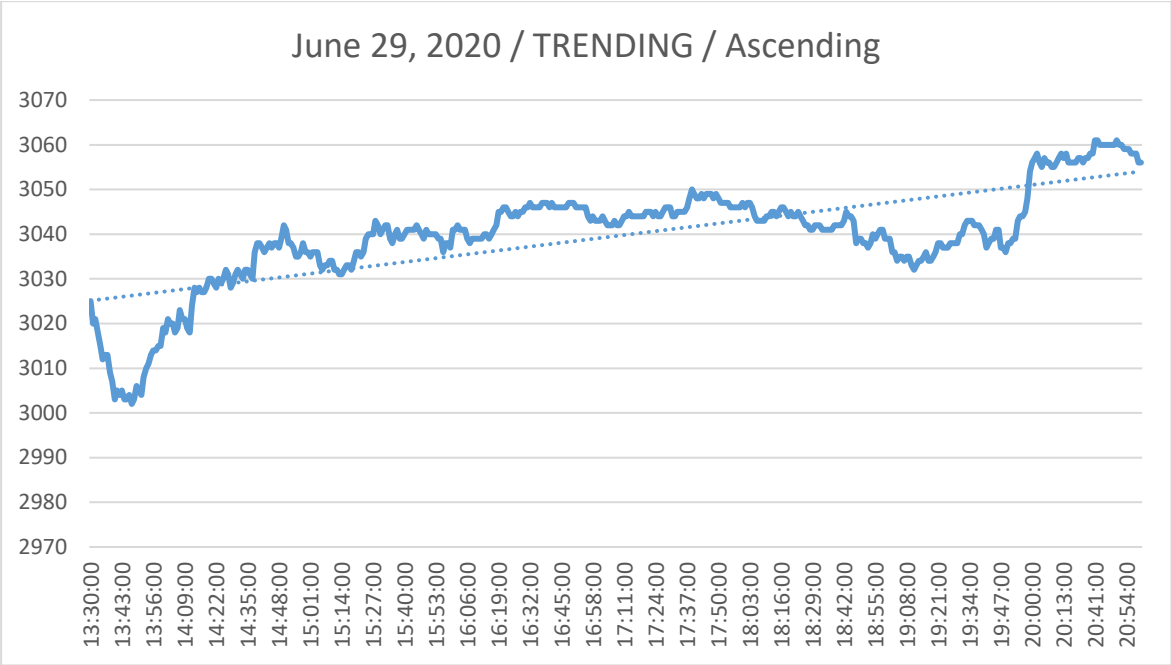
20:10:00	3371	3370,8	3370,7	3370,7	3370,8	3370,7
20:11:00	3370	3370,8	3370,6	3370,7	3370,5	3370,8
20:12:00	3368	3370,2	3370,3	3370,5	3369,7	3370,5
20:13:00	3368	3369,6	3370,0	3370,3	3369,5	3369,7
20:14:00	3369	3369,2	3369,9	3370,1	3369,6	3369,5
20:15:00	3369	3368,8	3369,8	3370,1	3369,4	3369,6
20:16:00	3369	3368,6	3369,7	3369,9	3369,3	3369,4
20:17:00	3368	3368,6	3369,4	3369,7	3368,9	3369,3
20:18:00	3368	3368,6	3369,1	3369,5	3368,8	3368,9
20:19:00	3368	3368,4	3368,8	3369,4	3368,7	3368,8
20:20:00	3368	3368,2	3368,5	3369,3	3368,5	3368,7
20:21:00	3369	3368,2	3368,4	3369,2	3368,7	3368,5
20:22:00	3370	3368,6	3368,6	3369,1	3369,1	3368,7
20:23:00	3370	3369,0	3368,8	3369,1	3369,2	3369,1
20:24:00	3371	3369,6	3369,0	3369,1	3369,7	3369,2
20:25:00	3370	3370,0	3369,1	3369,0	3369,5	3369,7
20:26:00	3370	3370,2	3369,2	3369,0	3369,6	3369,5
20:27:00	3371	3370,4	3369,5	3369,2	3370,0	3369,6
20:28:00	3371	3370,6	3369,8	3369,4	3370,2	3370,0
20:29:00	3371	3370,6	3370,1	3369,5	3370,3	3370,2
20:30:00	3371	3370,8	3370,4	3369,7	3370,5	3370,3
20:31:00	3371	3371,0	3370,6	3369,8	3370,6	3370,5
20:32:00	3371	3371,0	3370,7	3370,0	3370,7	3370,6
20:33:00	3372	3371,2	3370,9	3370,3	3371,1	3370,7
20:34:00	3372	3371,4	3371,0	3370,5	3371,2	3371,1
20:35:00	3372	3371,6	3371,2	3370,8	3371,4	3371,2
20:36:00	3373	3372,0	3371,5	3371,1	3371,9	3371,4
20:37:00	3374	3372,6	3371,8	3371,3	3372,4	3371,9
20:38:00	3373	3372,8	3372,0	3371,5	3372,3	3372,4
20:39:00	3374	3373,2	3372,3	3371,7	3372,8	3372,3
20:40:00	3374	3373,6	3372,6	3372,0	3373,1	3372,8
20:41:00	3374	3373,8	3372,9	3372,3	3373,2	3373,1
20:42:00	3374	3373,8	3373,2	3372,5	3373,4	3373,2
20:43:00	3374	3374,0	3373,4	3372,7	3373,5	3373,4
20:44:00	3374	3374,0	3373,6	3372,9	3373,6	3373,5
20:45:00	3374	3374,0	3373,8	3373,1	3373,7	3373,6
20:46:00	3374	3374,0	3373,9	3373,3	3373,8	3373,7
20:47:00	3374	3374,0	3373,9	3373,5	3373,8	3373,8
20:48:00	3375	3374,2	3374,1	3373,7	3374,2	3373,8
20:49:00	3376	3374,6	3374,3	3373,9	3374,7	3374,2
20:50:00	3376	3375,0	3374,5	3374,2	3374,9	3374,7
20:51:00	3376	3375,4	3374,7	3374,4	3375,1	3374,9
20:52:00	3376	3375,8	3374,9	3374,5	3375,3	3375,1
20:53:00	3377	3376,2	3375,2	3374,8	3375,8	3375,3
20:54:00	3377	3376,4	3375,5	3375,0	3376,0	3375,8

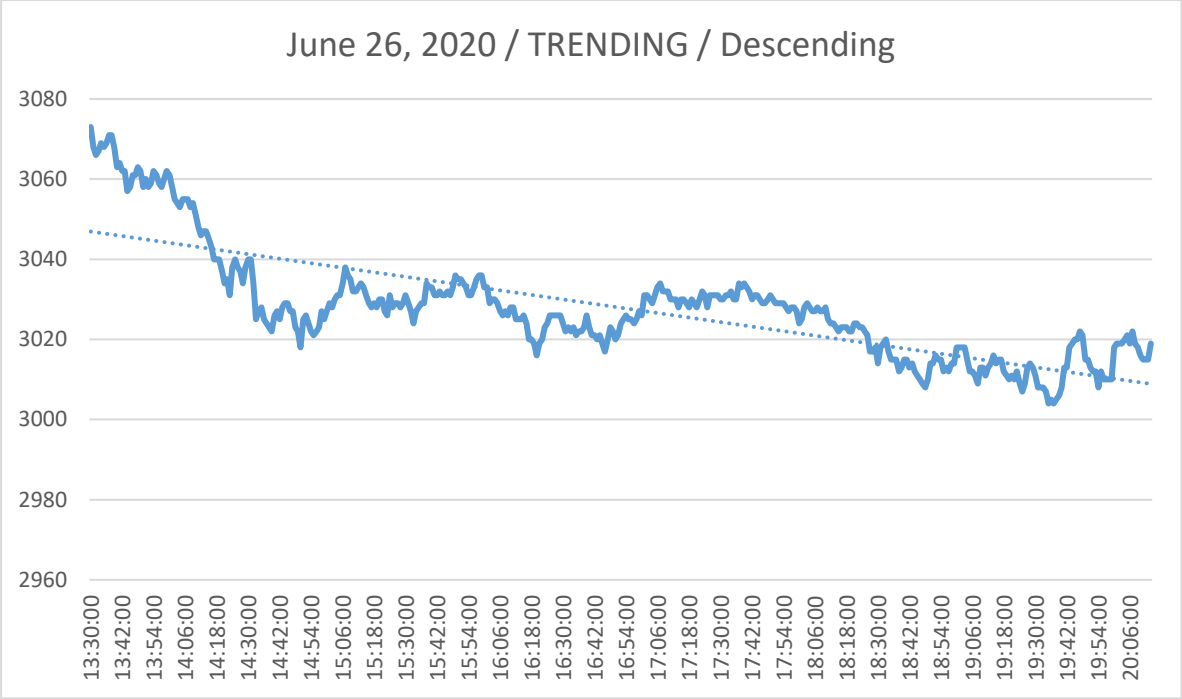
20:55:00	3377	3376,6	3375,8	3375,2	3376,2	3376,0
20:56:00	3378	3377,0	3376,2	3375,5	3376,7	3376,2
20:57:00	3377	3377,2	3376,5	3375,7	3376,6	3376,7
20:58:00	3378	3377,4	3376,8	3375,9	3377,0	3376,6
20:59:00	3378	3377,6	3377,0	3376,2	3377,2	3377,0

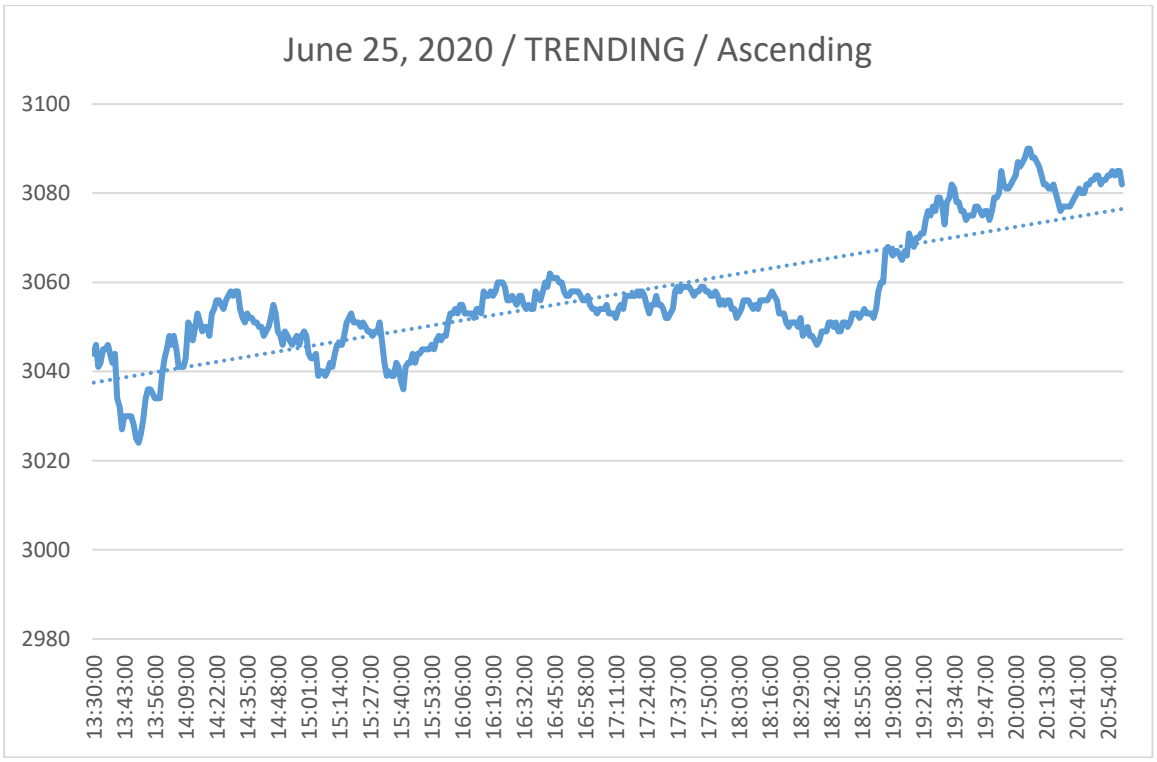
APPENDIX II

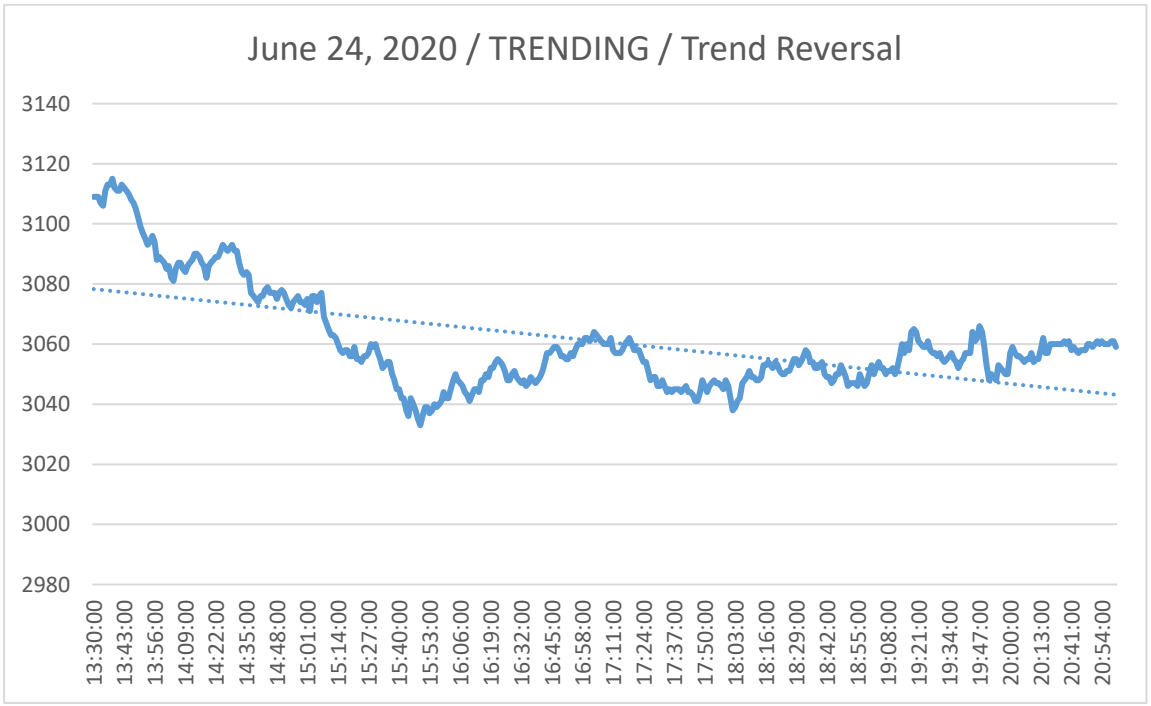
TRADING DAYS for the first half 2020

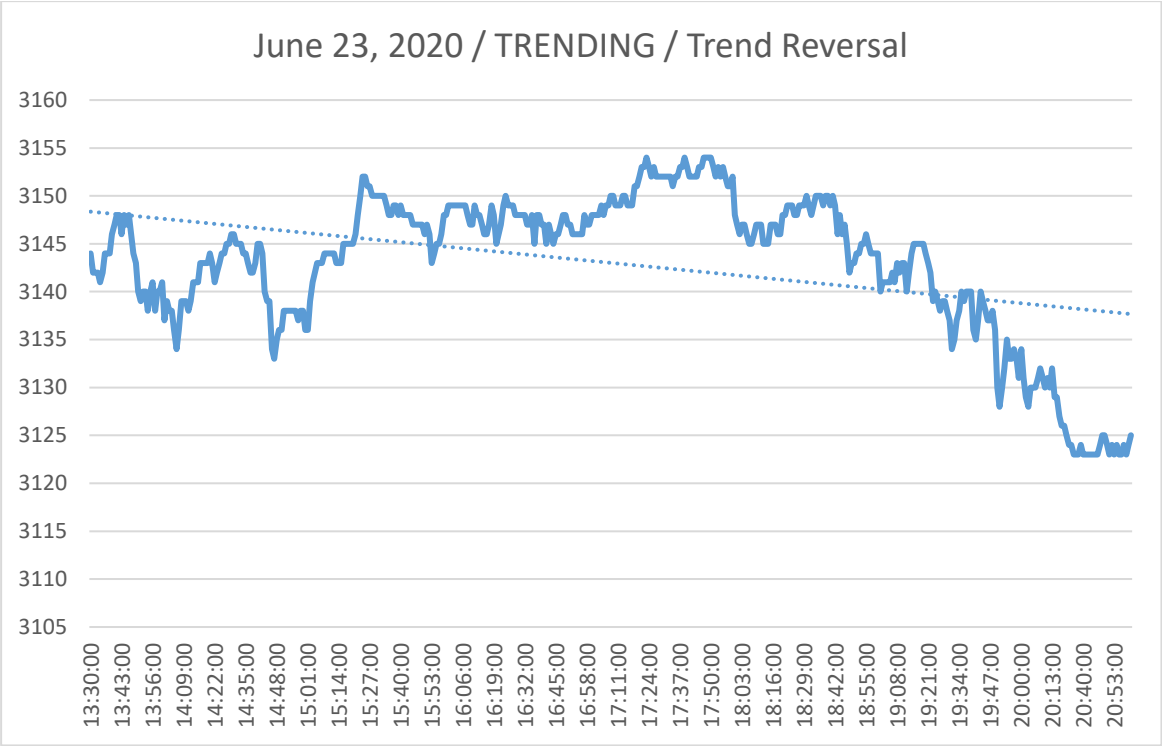


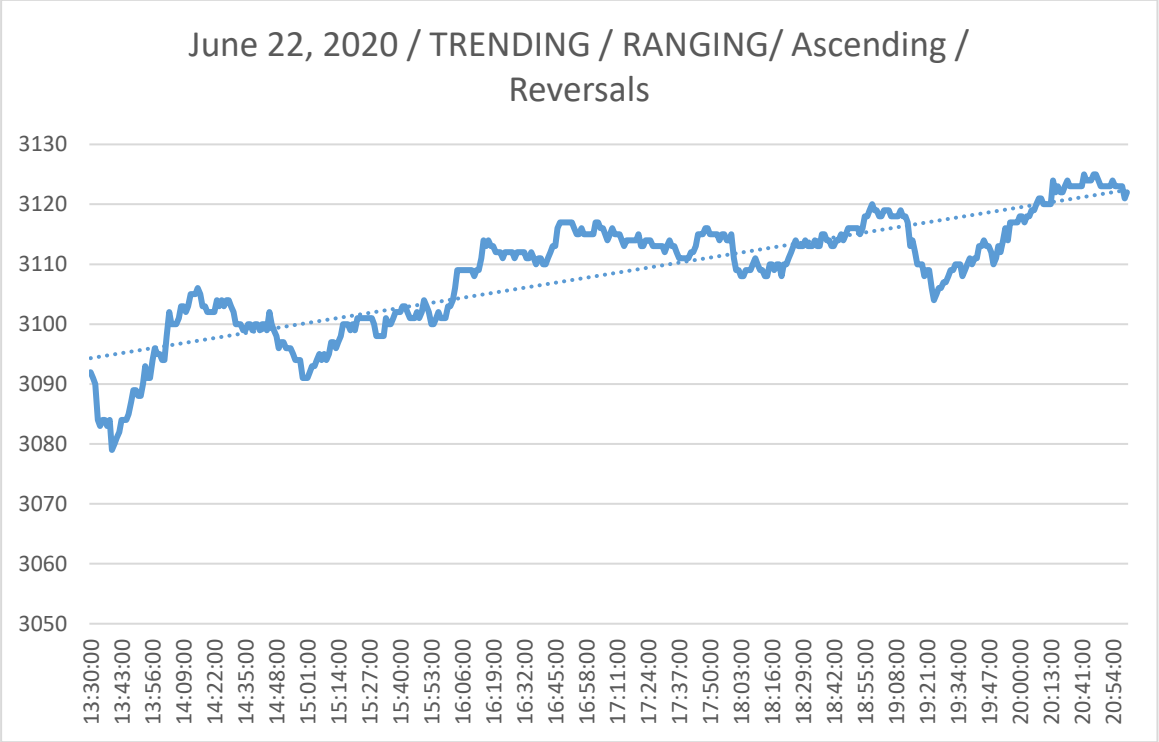

June 29, 2020 / TRENDING / Ascending
3070
3060
3050
3040
3030
3020
3010
3000
2990
2980
2970
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

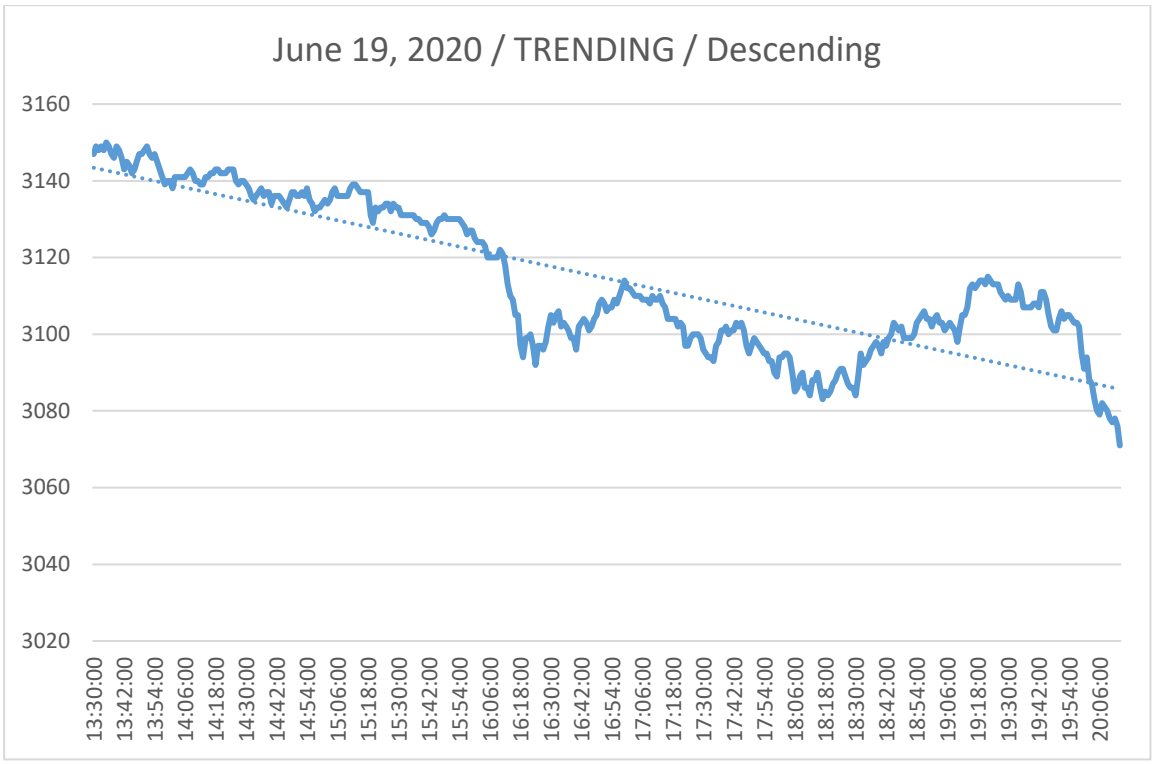

June 26, 2020 / TRENDING / Descending
3080
3060
3040
3020
3000
2980
2960
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:30:00
14:42:00
14:54:00
15:06:00
15:18:00
15:30:00
15:42:00
15:54:00
16:06:00
16:18:00
16:30:00
16:42:00
16:54:00
17:06:00
17:18:00
17:30:00
17:42:00
17:54:00
18:06:00
18:18:00
18:30:00
18:42:00
18:54:00
19:06:00
19:18:00
19:30:00
19:42:00
19:54:00
20:06:00

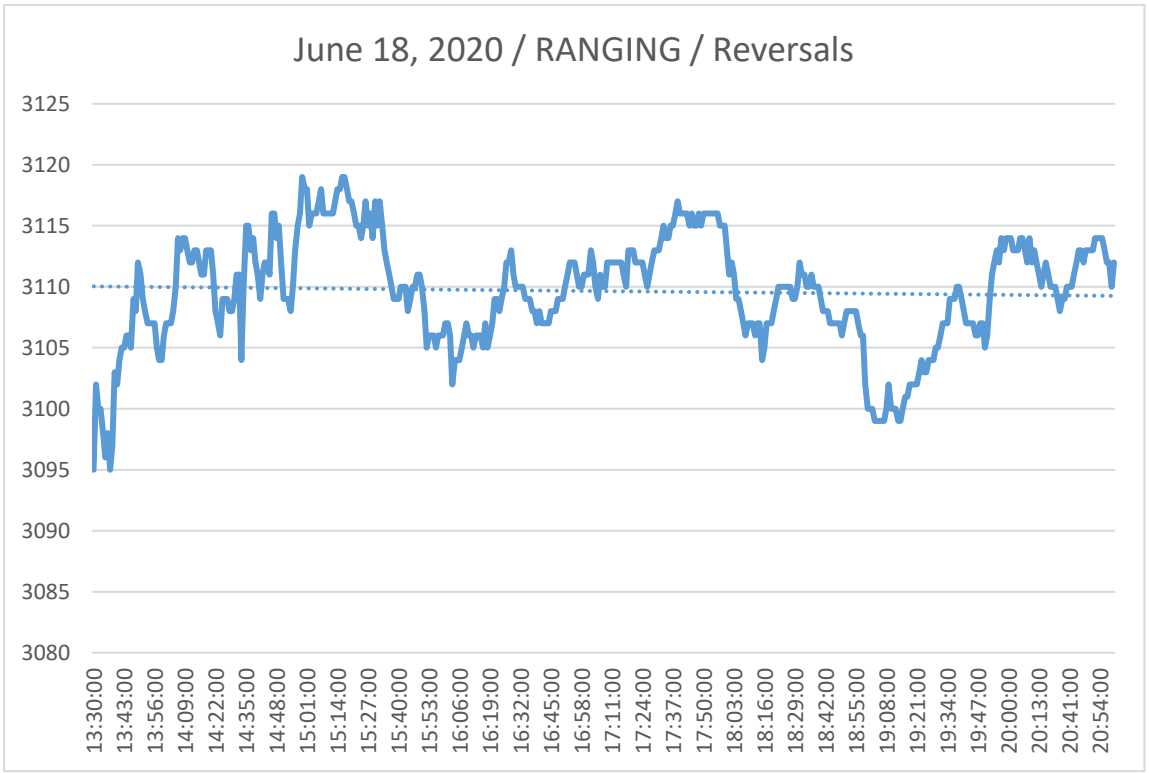

June 25, 2020 / TRENDING / Ascending
3100
3080
3060
3040
3020
3000
2980
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

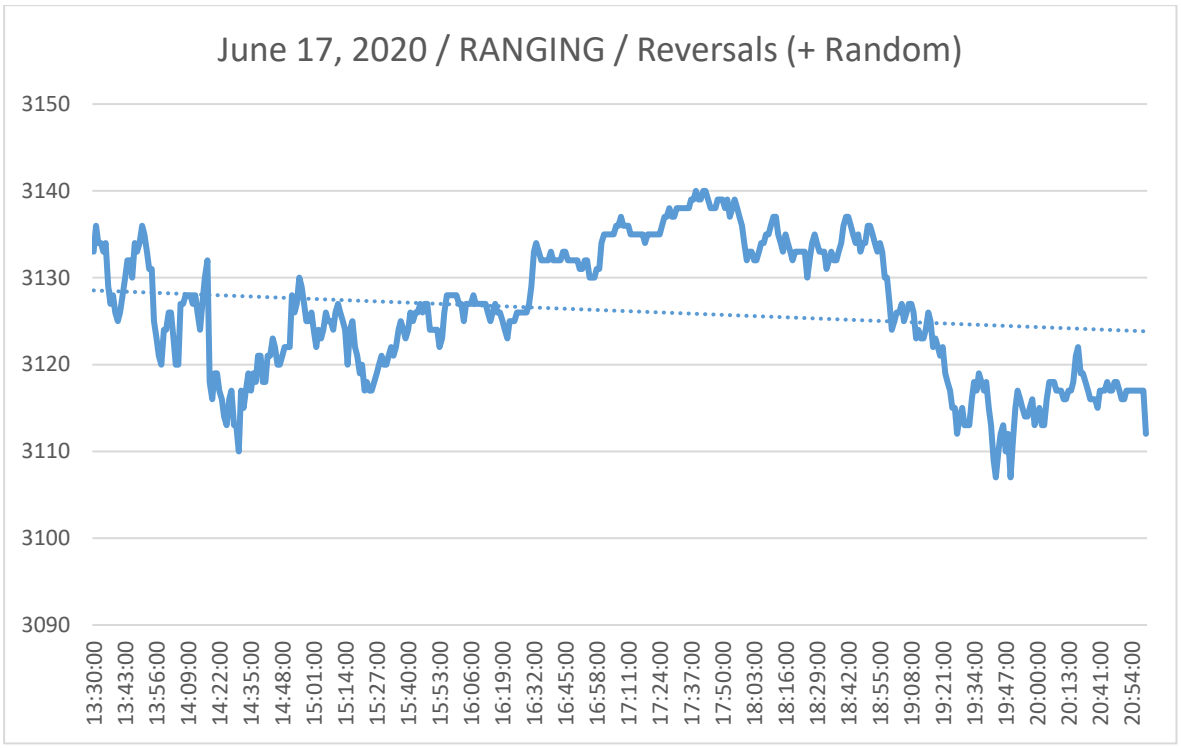

June 24, 2020 / TRENDING / Trend Reversal
3140
3120
3100
3080
3060
3040
3020
3000
2980
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

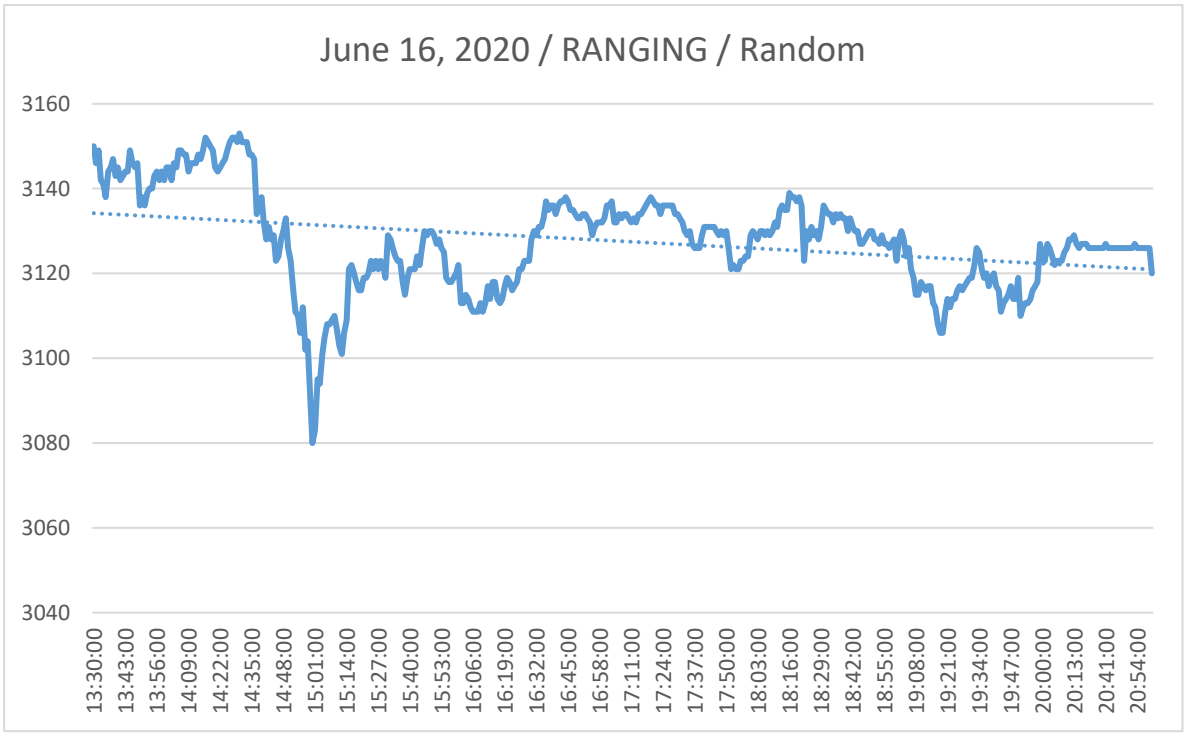

June 23, 2020 / TRENDING / Trend Reversal
3160
3155
3150
3145
3140
3135
3130
3125
3120
3115
3110
3105
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:40:00
20:53:00

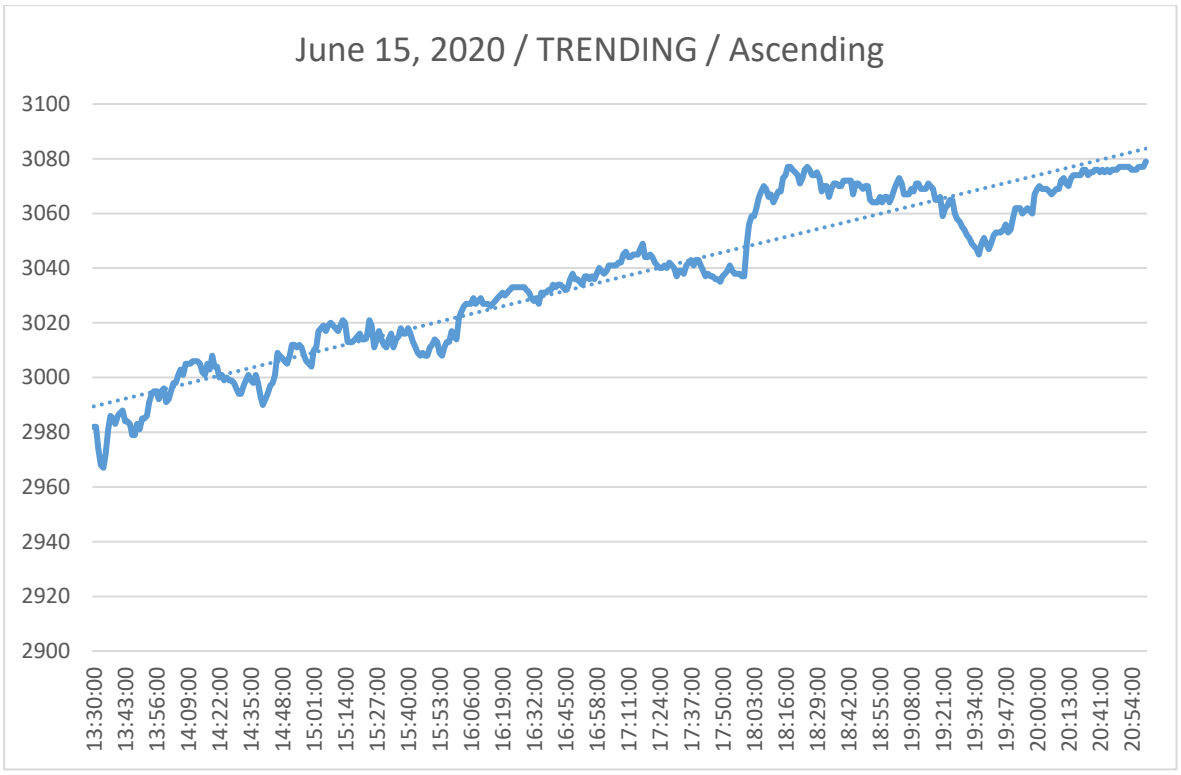

June 22, 2020 / TRENDING / RANGING/ Ascending / Reversals
3130
3120
3110
3100
3090
3080
3070
3060
3050
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

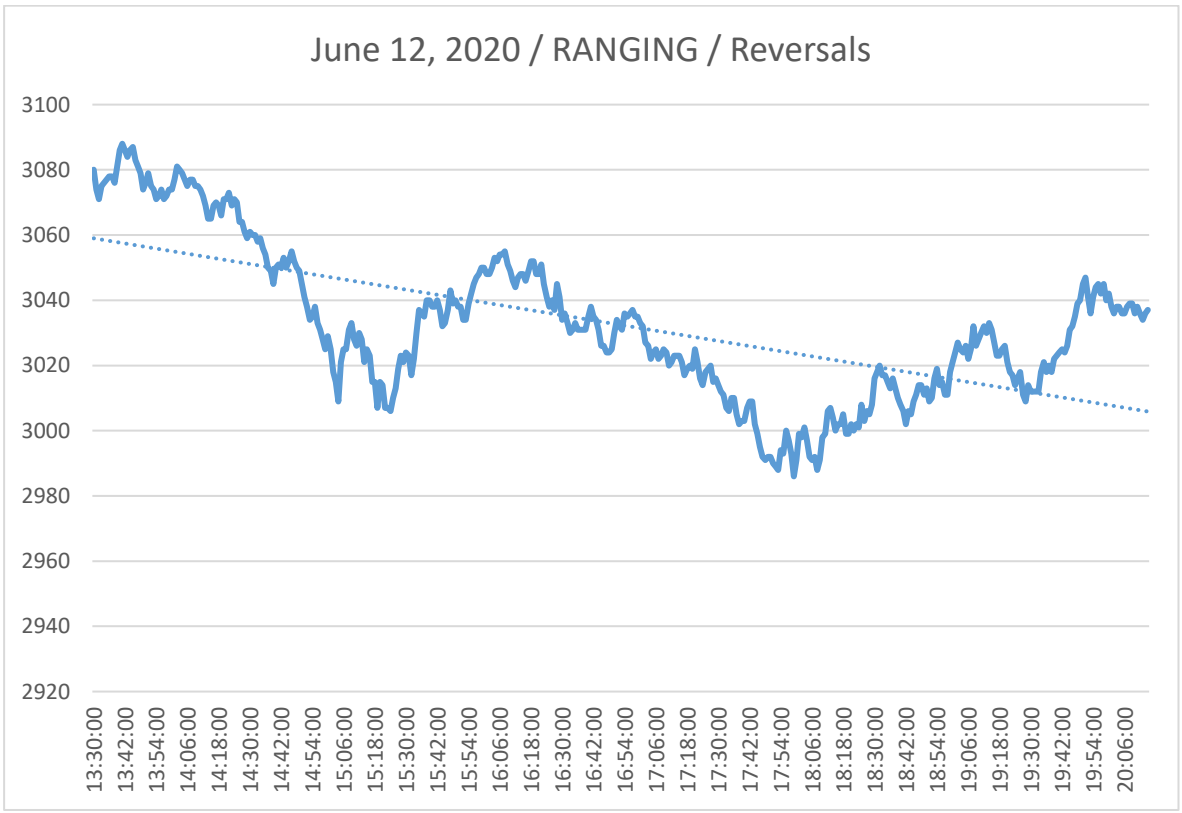

June 19, 2020 / TRENDING / Descending
3160
3140
3120
3100
3080
3060
3040
3020
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:30:00
14:42:00
14:54:00
15:06:00
15:18:00
15:30:00
15:42:00
15:54:00
16:06:00
16:18:00
16:30:00
16:42:00
16:54:00
17:06:00
17:18:00
17:30:00
17:42:00
17:54:00
18:06:00
18:18:00
18:30:00
18:42:00
18:54:00
19:06:00
19:18:00
19:30:00
19:42:00
19:54:00
20:06:00

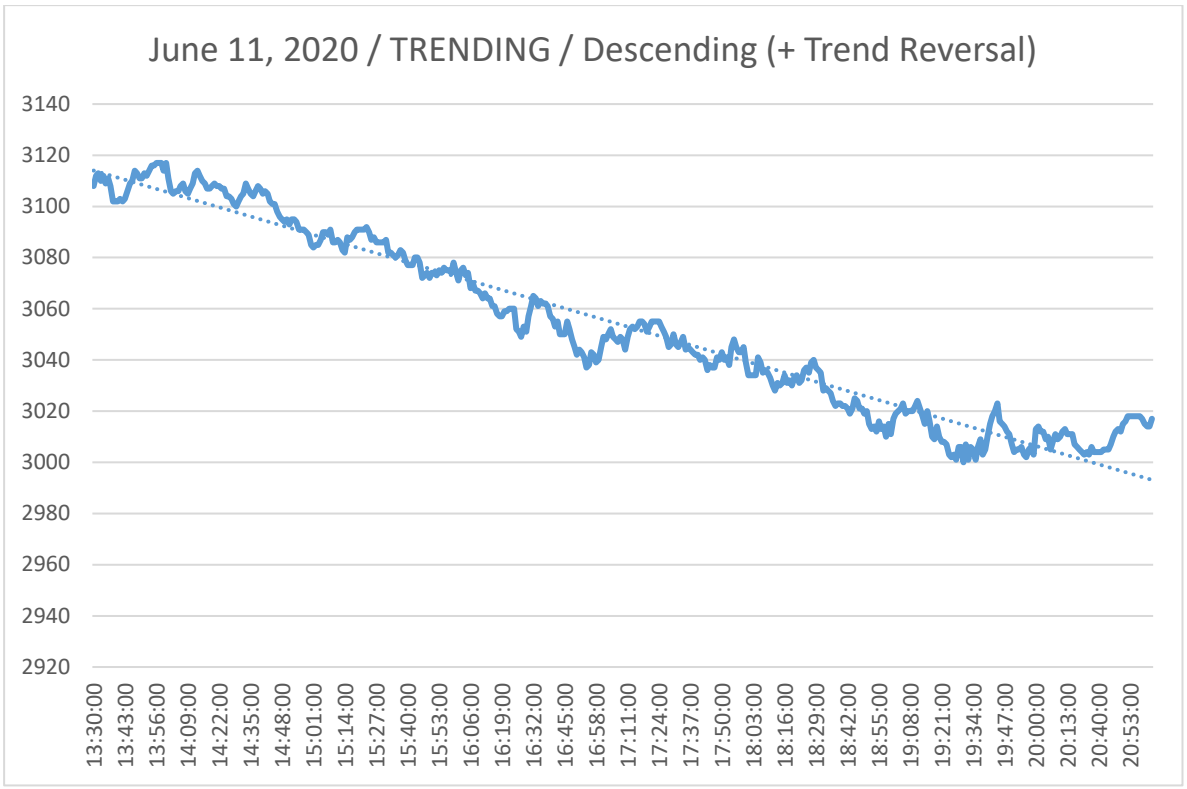

June 18, 2020 / RANGING / Reversals
3125
3120
3115
3110
3105
3100
3095
3090
3085
3080
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

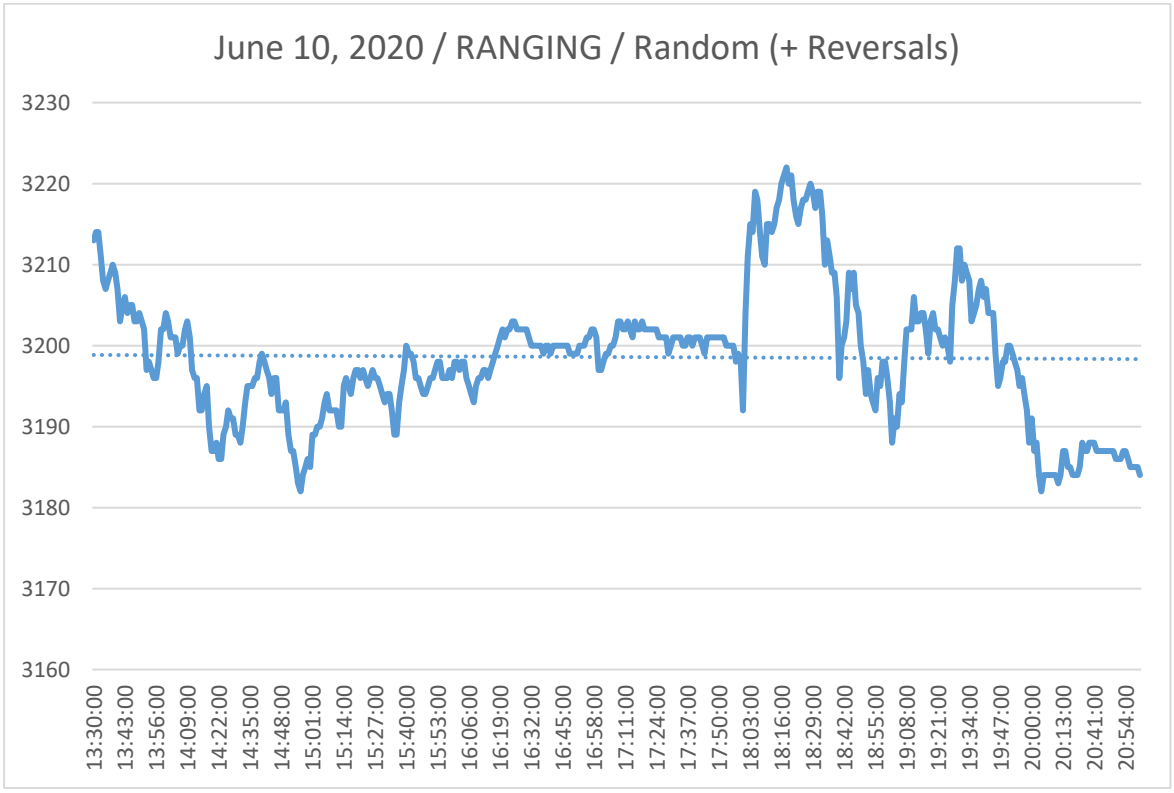

June 17, 2020 / RANGING / Reversals (+ Random)
3150
3140
3130
3120
3110
3100
3090
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

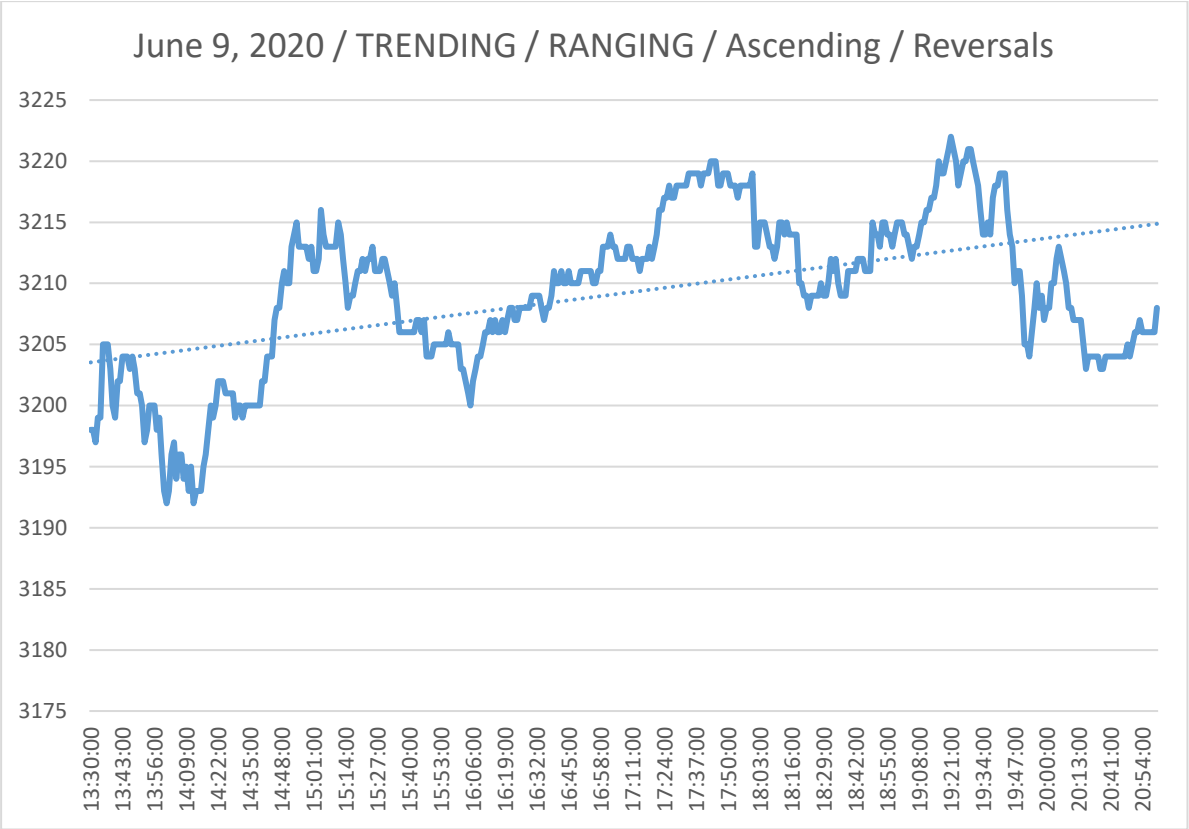

June 16, 2020 / RANGING / Random
3160
3140
3120
3100
3080
3060
3040
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

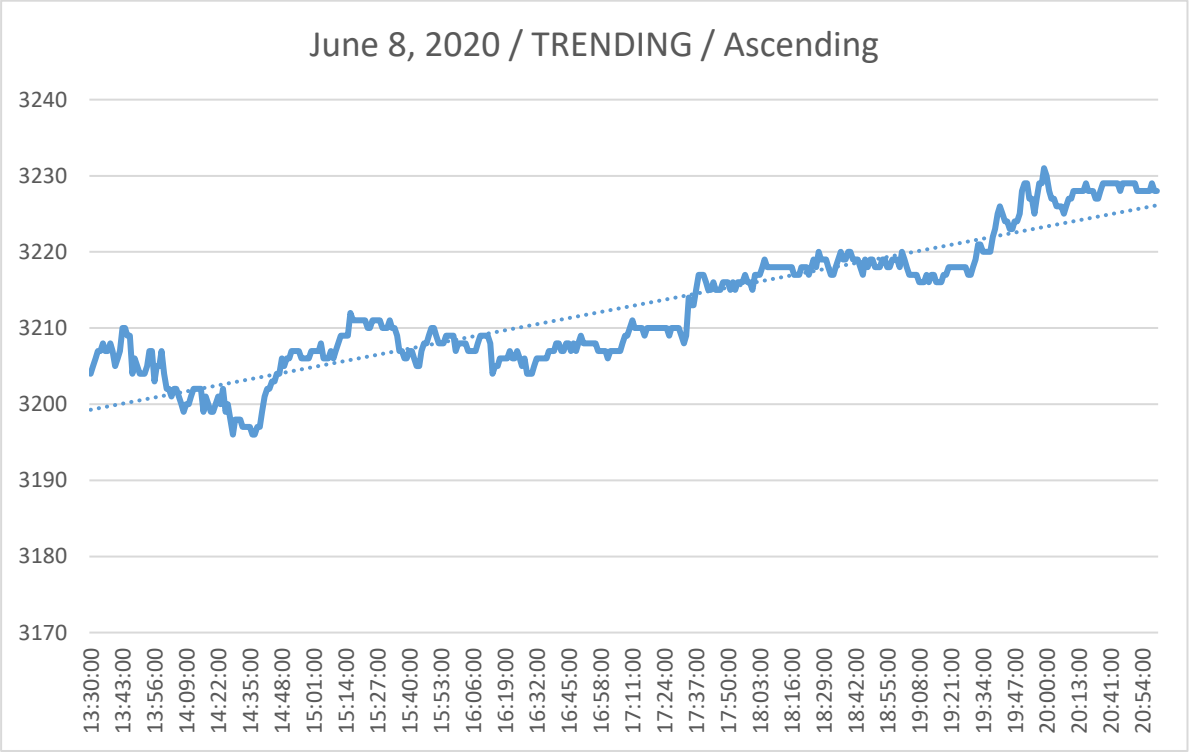

June 15, 2020 / TRENDING / Ascending
3100
3080
3060
3040
3020
3000
2980
2960
2940
2920
2900
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

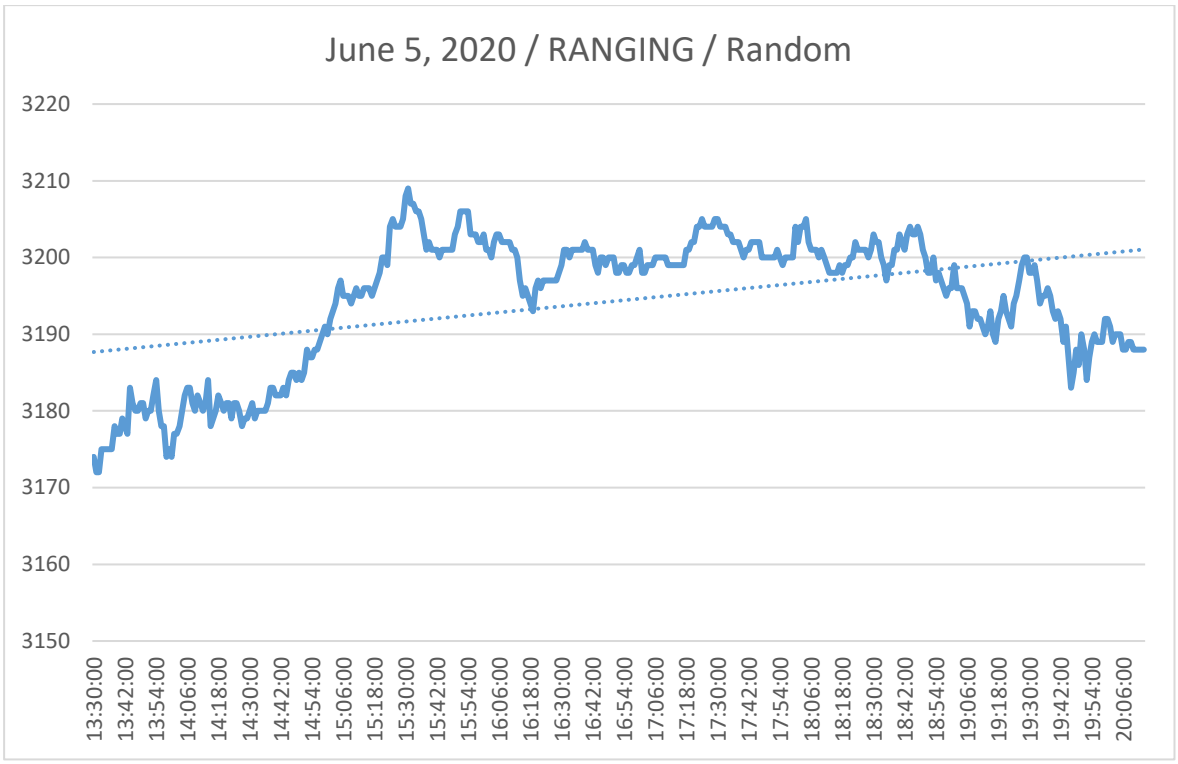

June 12, 2020 / RANGING / Reversals
3100
3080
3060
3040
3020
3000
2980
2960
2940
2920
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:30:00
14:42:00
14:54:00
15:06:00
15:18:00
15:30:00
15:42:00
15:54:00
16:06:00
16:18:00
16:30:00
16:42:00
16:54:00
17:06:00
17:18:00
17:30:00
17:42:00
17:54:00
18:06:00
18:18:00
18:30:00
18:42:00
18:54:00
19:06:00
19:18:00
19:30:00
19:42:00
19:54:00
20:06:00

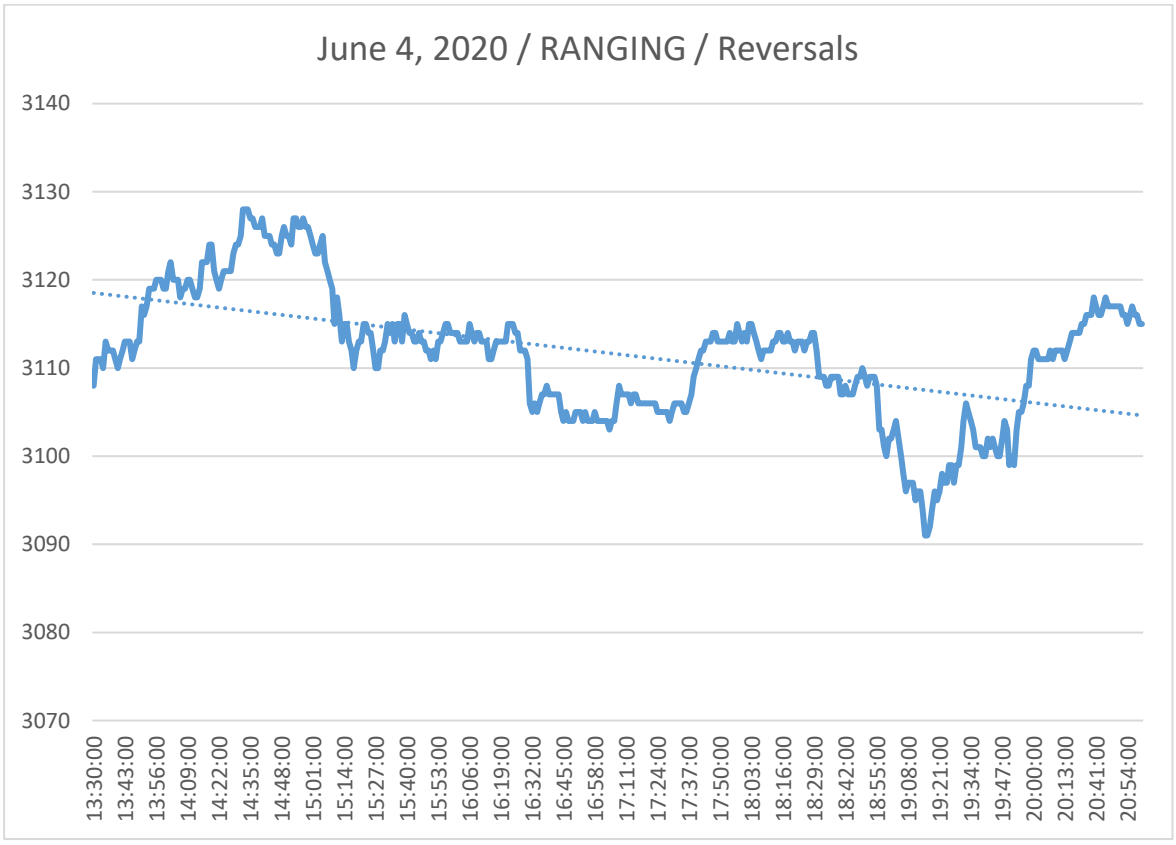

June 11, 2020 / TRENDING / Descending (+ Trend Reversal)
3140
3120
3100
3080
3060
3040
3020
3000
2980
2960
2940
2920
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:40:00
20:53:00

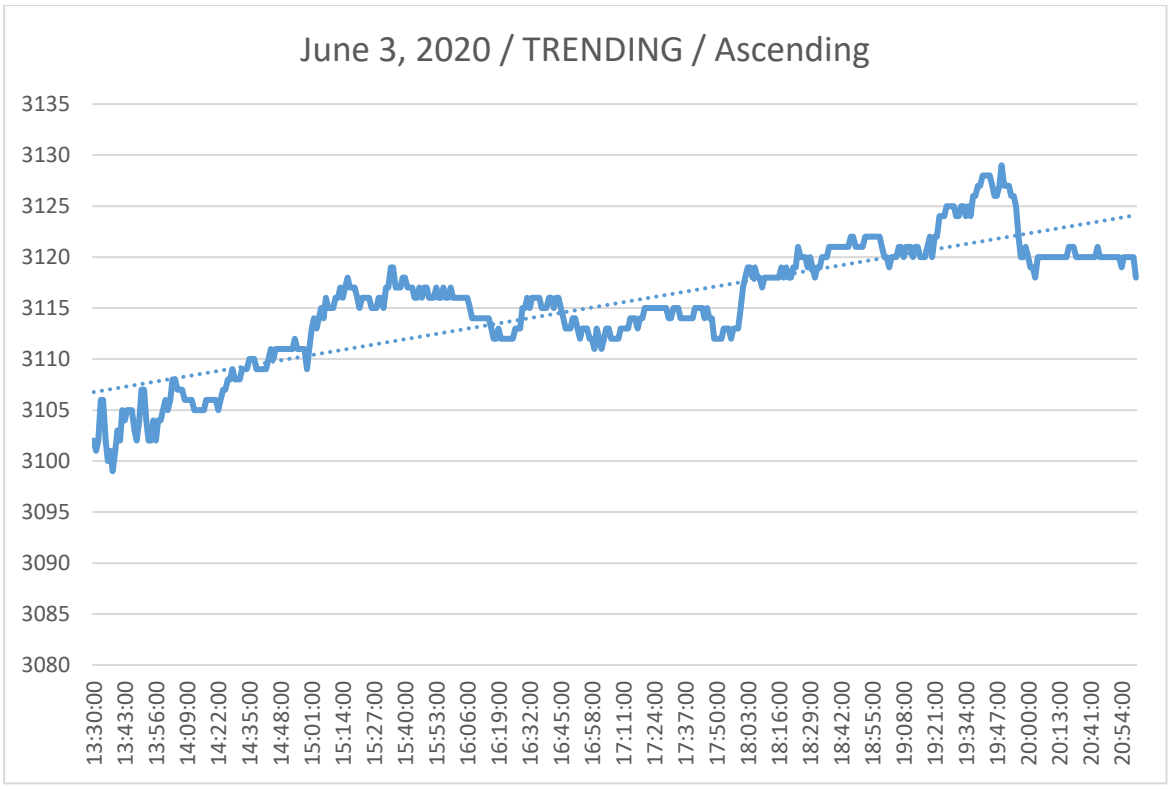

June 10, 2020 / RANGING / Random (+ Reversals)
3230
3220
3210
3200
3190
3180
3170
3160
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

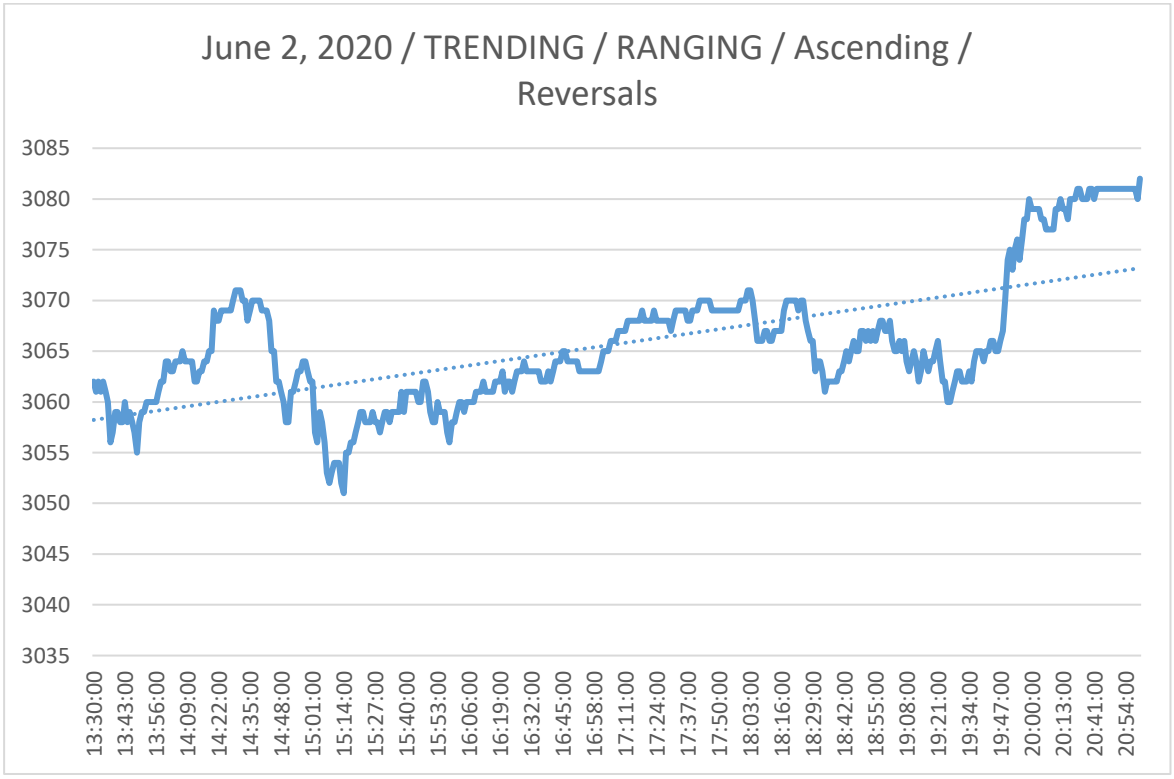

June 9, 2020 / TRENDING / RANGING / Ascending / Reversals
3225
3220
3215
3210
3205
3200
3195
3190
3185
3180
3175
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

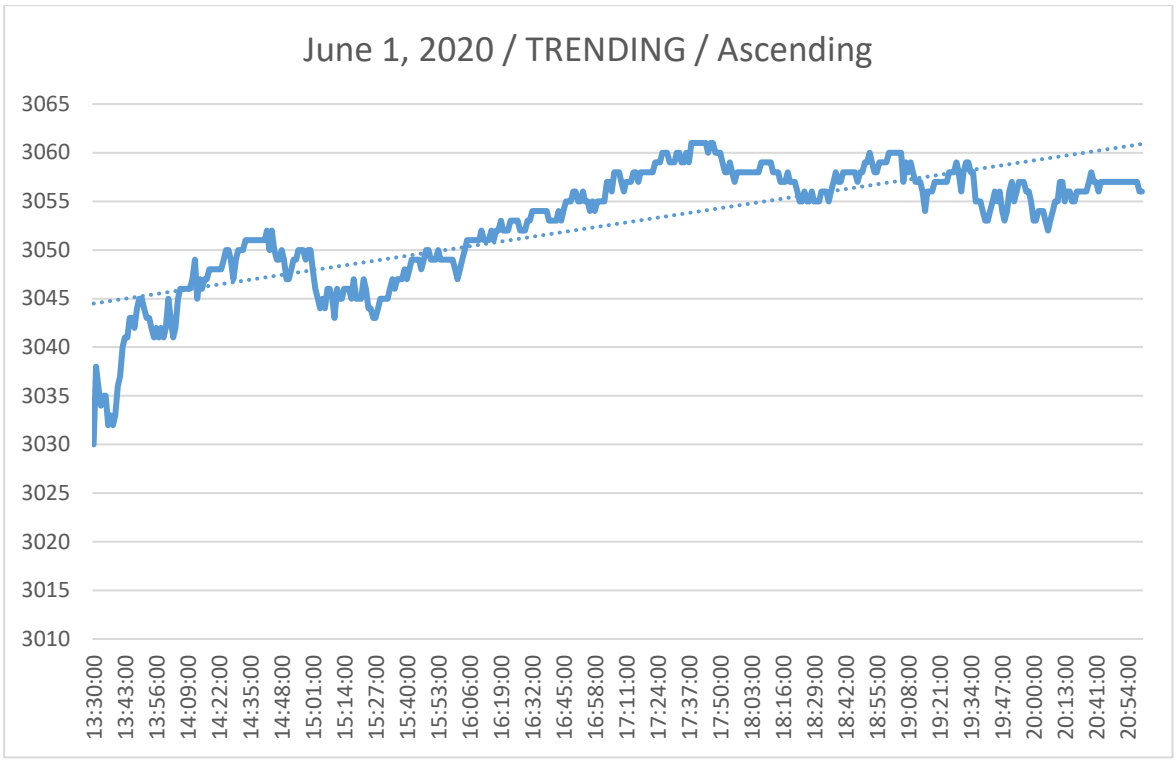

June 8, 2020 / TRENDING / Ascending
3240
3230
3220
3210
3200
3190
3180
3170
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

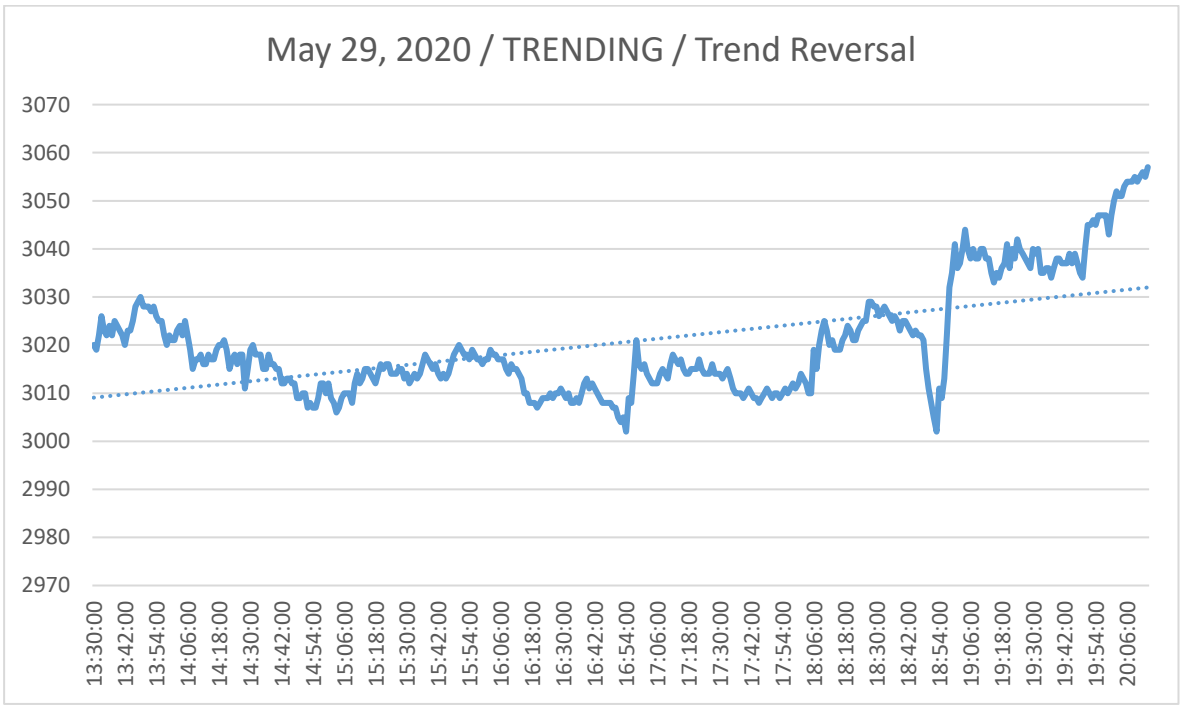

June 5, 2020 / RANGING / Random
3220
3210
3200
3190
3180
3170
3160
3150
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:30:00
14:42:00
14:54:00
15:06:00
15:18:00
15:30:00
15:42:00
15:54:00
16:06:00
16:18:00
16:30:00
16:42:00
16:54:00
17:06:00
17:18:00
17:30:00
17:42:00
17:54:00
18:06:00
18:18:00
18:30:00
18:42:00
18:54:00
19:06:00
19:18:00
19:30:00
19:42:00
19:54:00
20:06:00

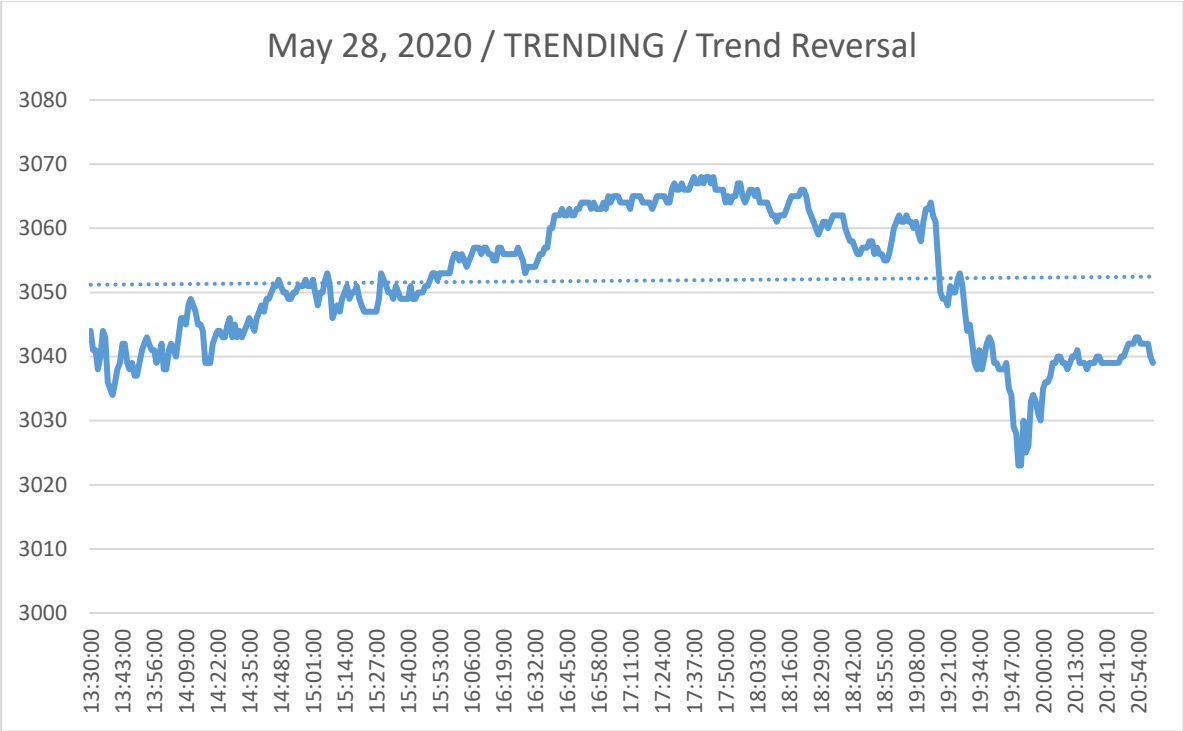

June 4, 2020 / RANGING / Reversals
3140
3130
3120
3110
3100
3090
3080
3070
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

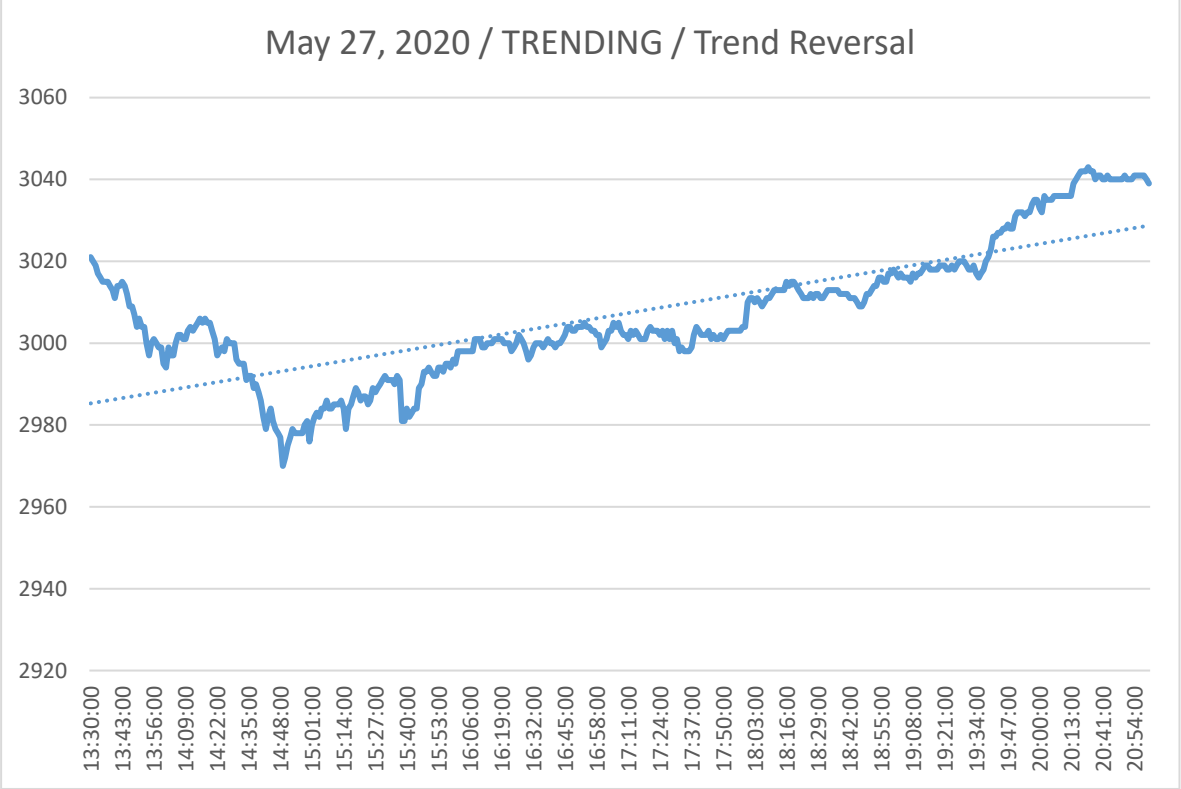

June 3, 2020 / TRENDING / Ascending
3135
3130
3125
3120
3115
3110
3105
3100
3095
3090
3085
3080
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

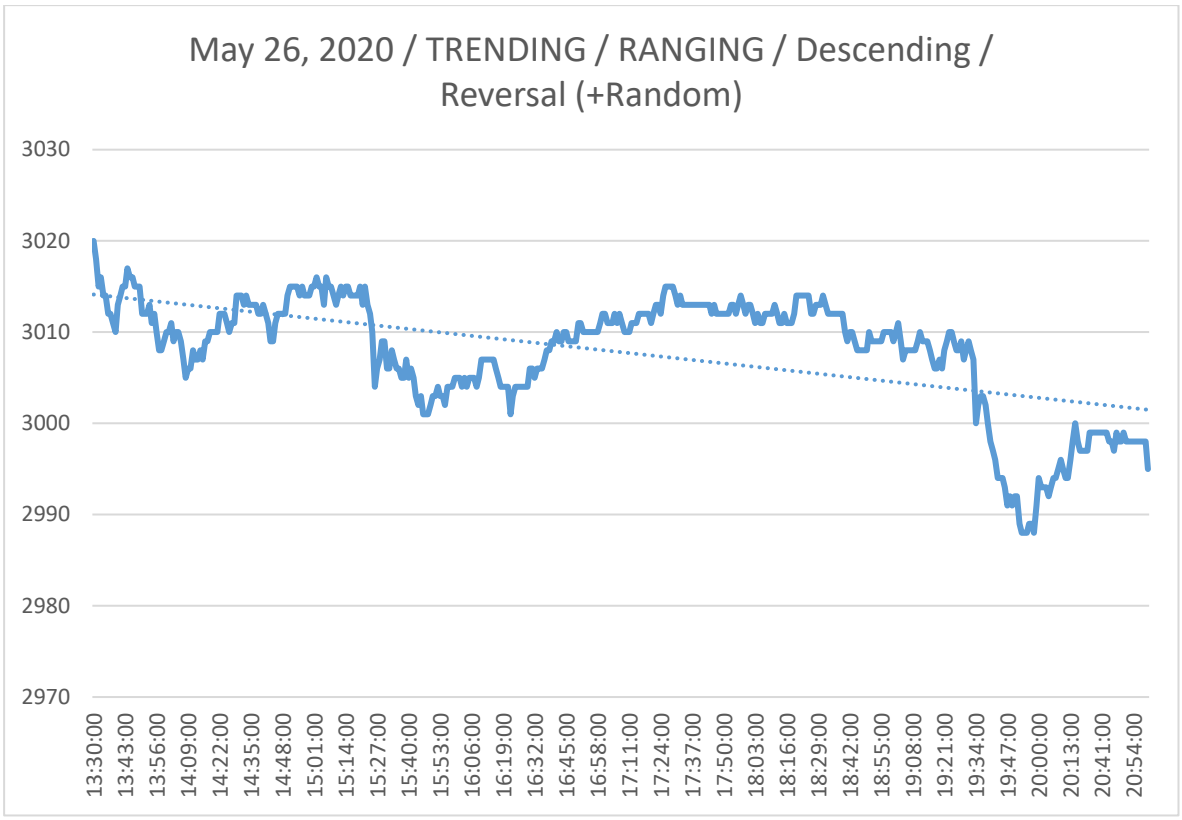

June 2, 2020 / TRENDING / RANGING / Ascending / Reversals
3085
3080
3075
3070
3065
3060
3055
3050
3045
3040
3035
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

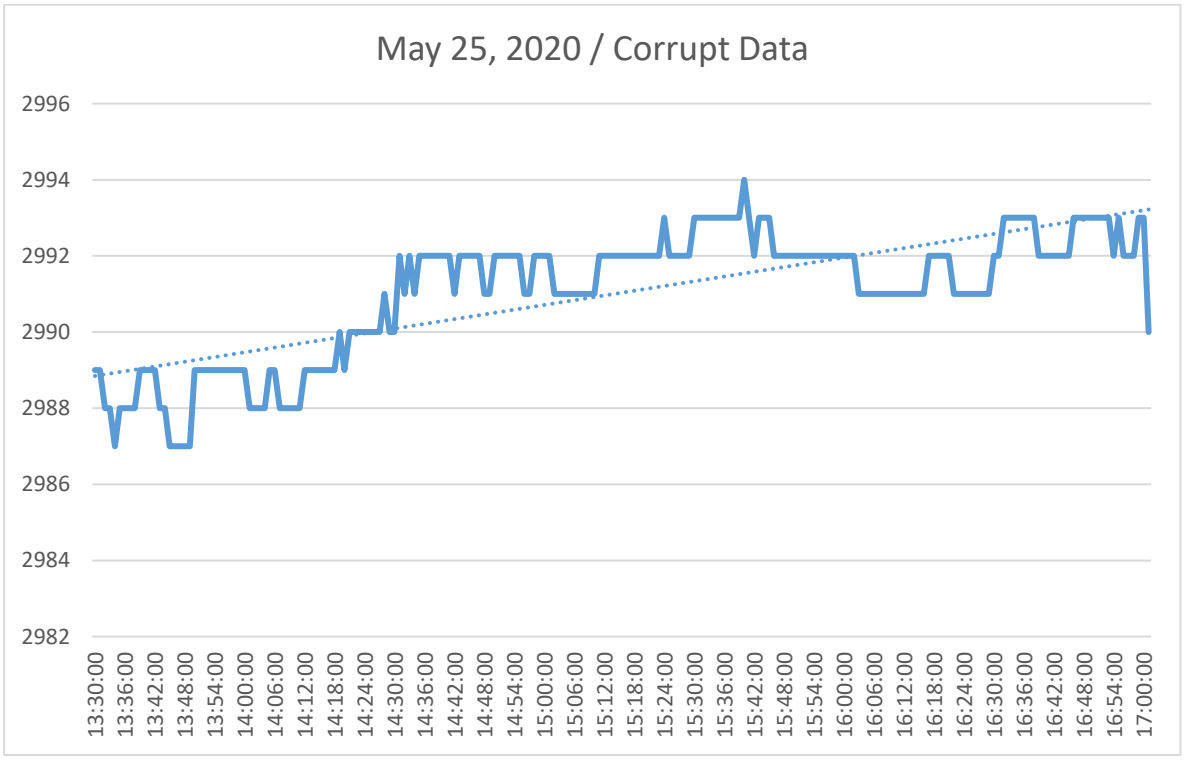

June 1, 2020 / TRENDING / Ascending
3065
3060
3055
3050
3045
3040
3035
3030
3025
3020
3015
3010
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

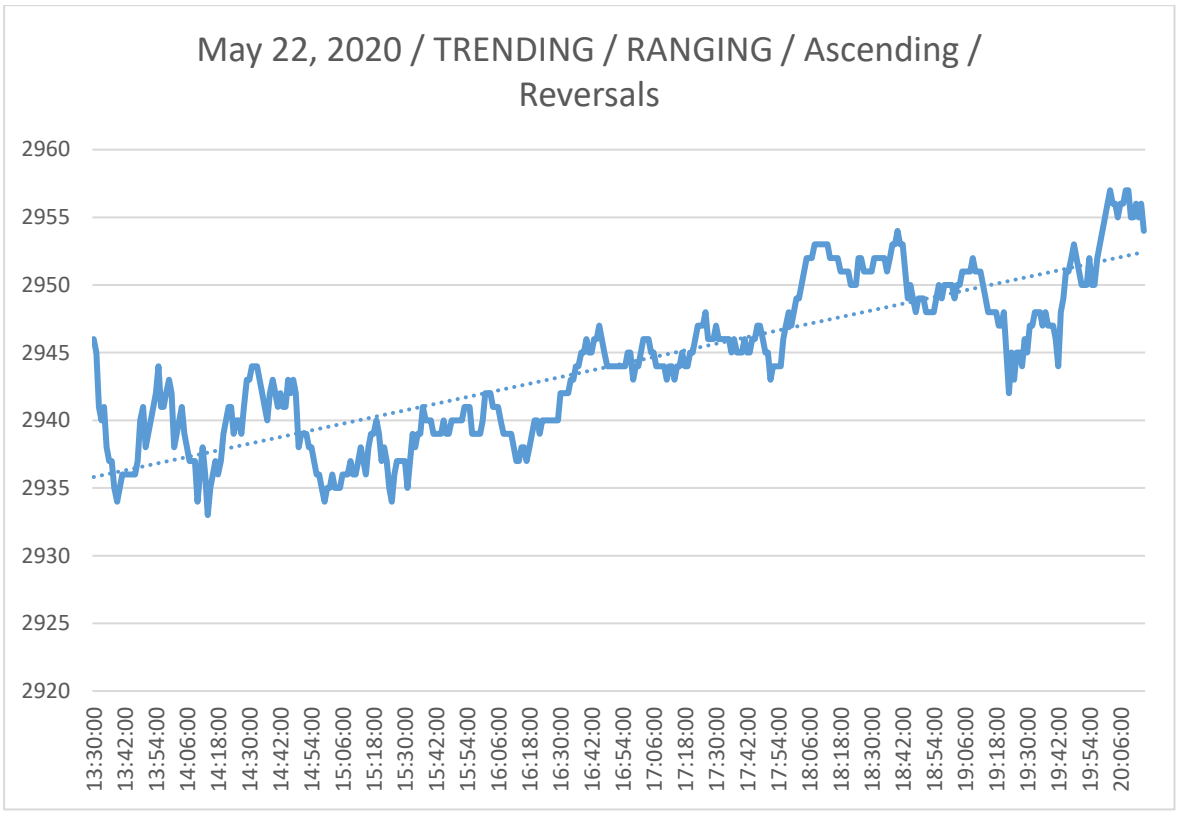

May 29, 2020 / TRENDING / Trend Reversal
3070
3060
3050
3040
3030
3020
3010
3000
2990
2980
2970
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:30:00
14:42:00
14:54:00
15:06:00
15:18:00
15:30:00
15:42:00
15:54:00
16:06:00
16:18:00
16:30:00
16:42:00
16:54:00
17:06:00
17:18:00
17:30:00
17:42:00
17:54:00
18:06:00
18:18:00
18:30:00
18:42:00
18:54:00
19:06:00
19:18:00
19:30:00
19:42:00
19:54:00
20:06:00

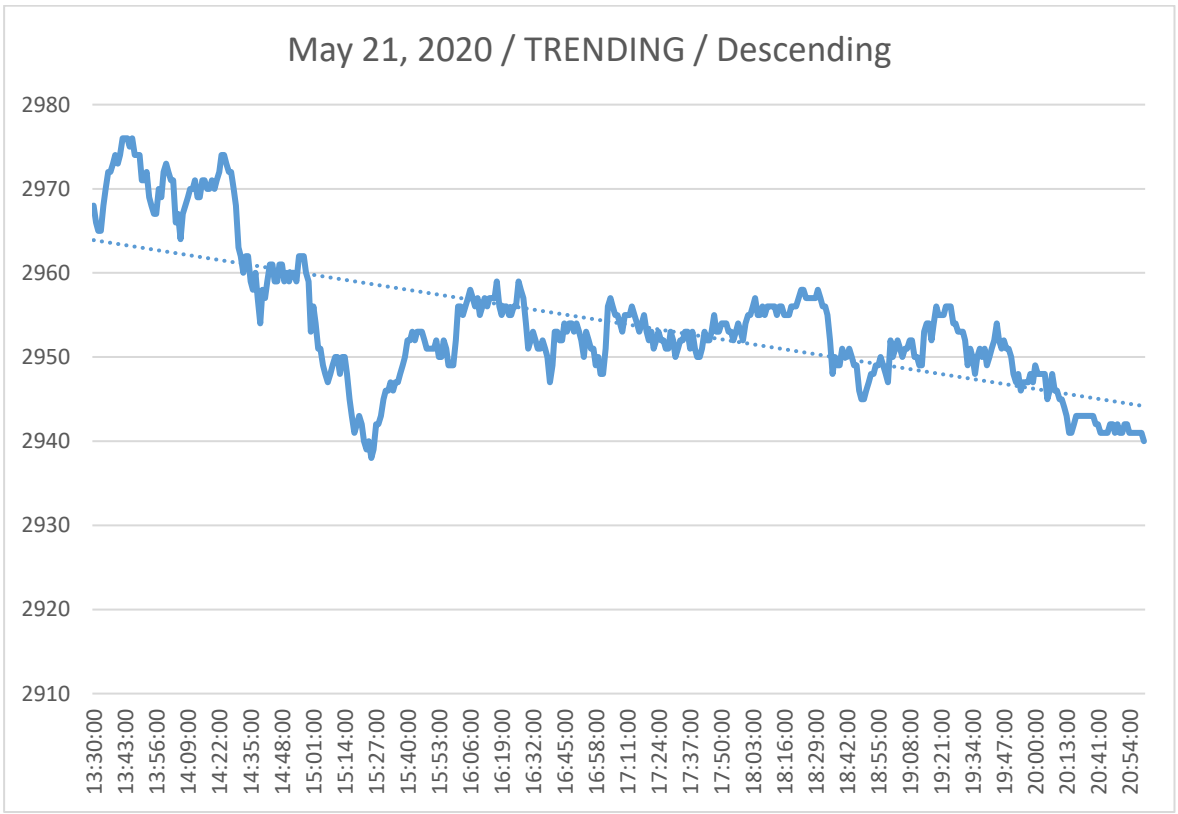

May 28, 2020 / TRENDING / Trend Reversal
3080
3070
3060
3050
3040
3030
3020
3010
3000
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

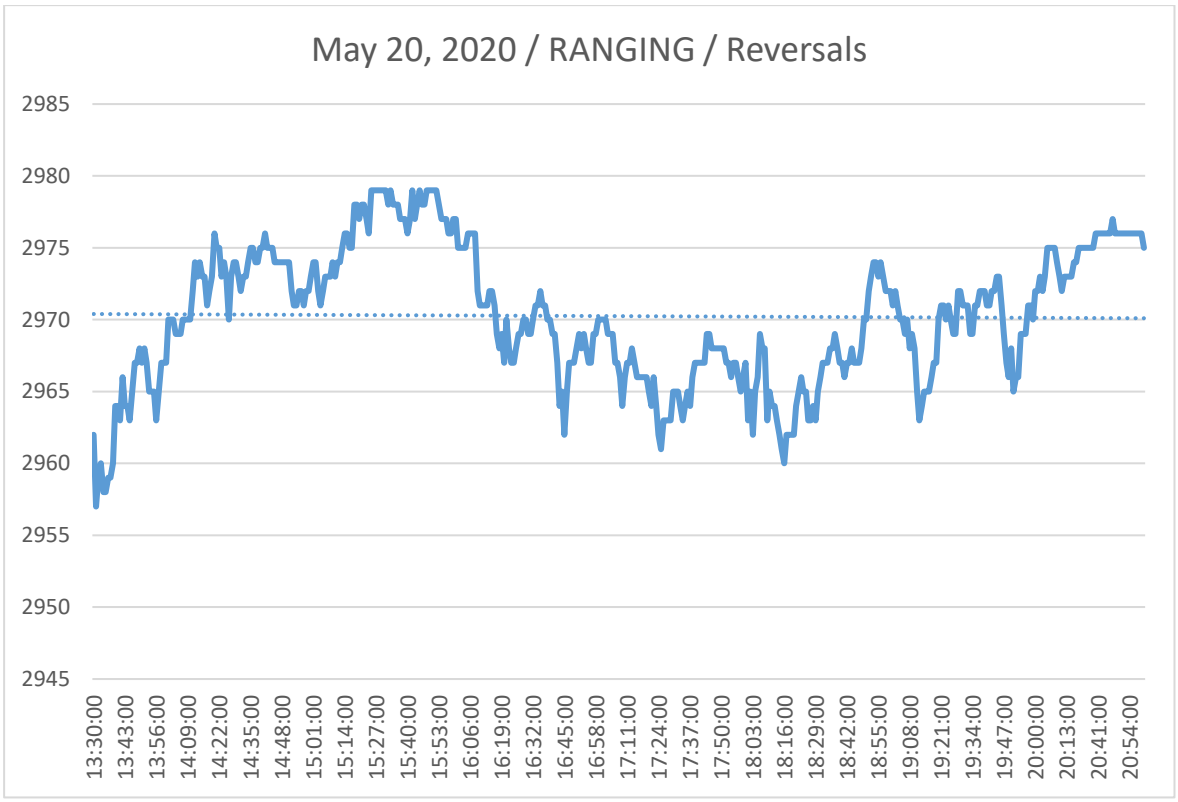

May 27, 2020 / TRENDING / Trend Reversal
3060
3040
3020
3000
2980
2960
2940
2920
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

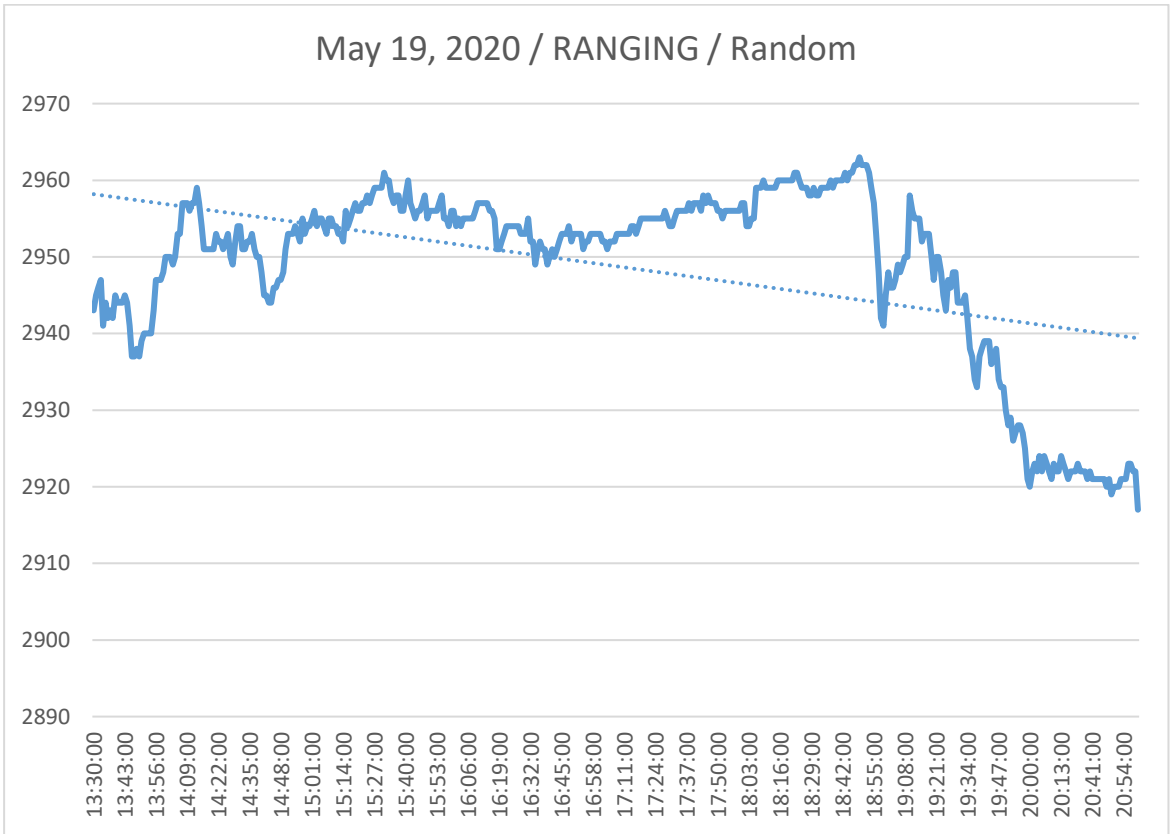

May 26, 2020 / TRENDING / RANGING / Descending / Reversal (+Random)
3030
3020
3010
3000
2990
2980
2970
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

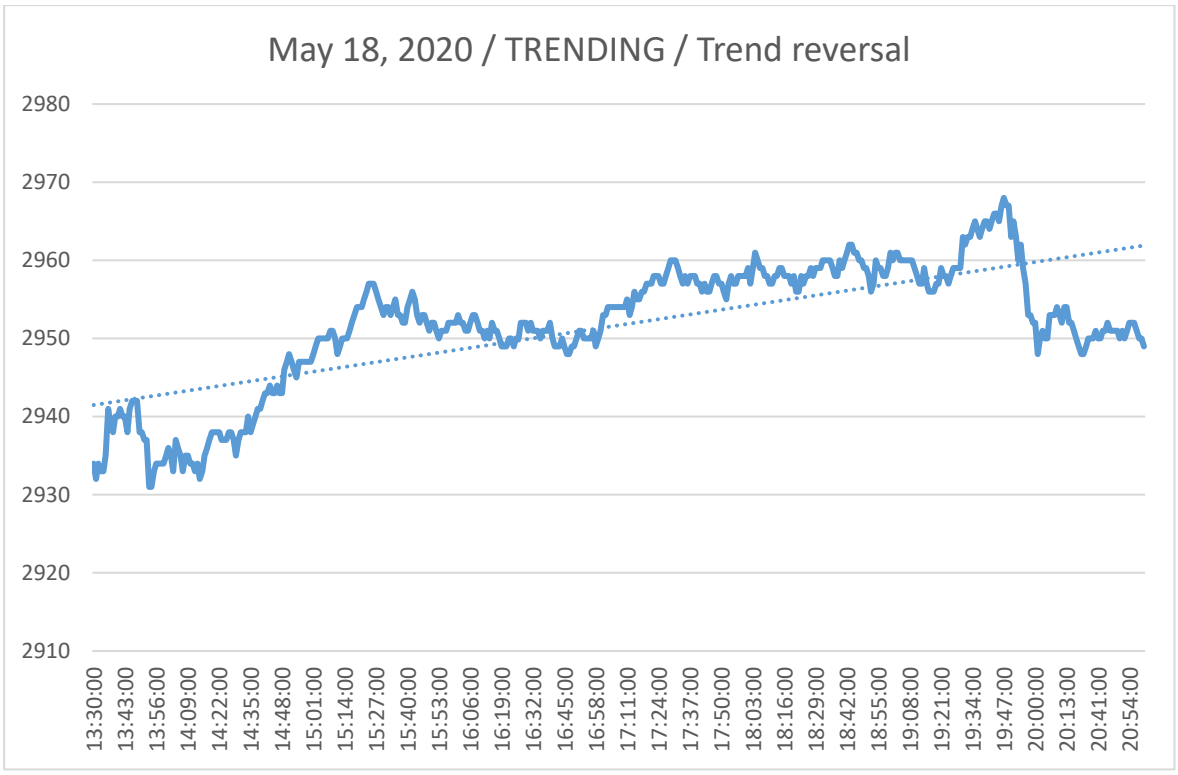

May 25, 2020 / Corrupt Data
2996
2994
2992
2990
2988
2986
2984
2982
13:30:00
13:36:00
13:42:00
13:48:00
13:54:00
14:00:00
14:06:00
14:12:00
14:18:00
14:24:00
14:30:00
14:36:00
14:42:00
14:48:00
14:54:00
15:00:00
15:06:00
15:12:00
15:18:00
15:24:00
15:30:00
15:36:00
15:42:00
15:48:00
15:54:00
16:00:00
16:06:00
16:12:00
16:18:00
16:24:00
16:30:00
16:36:00
16:42:00
16:48:00
16:54:00
17:00:00

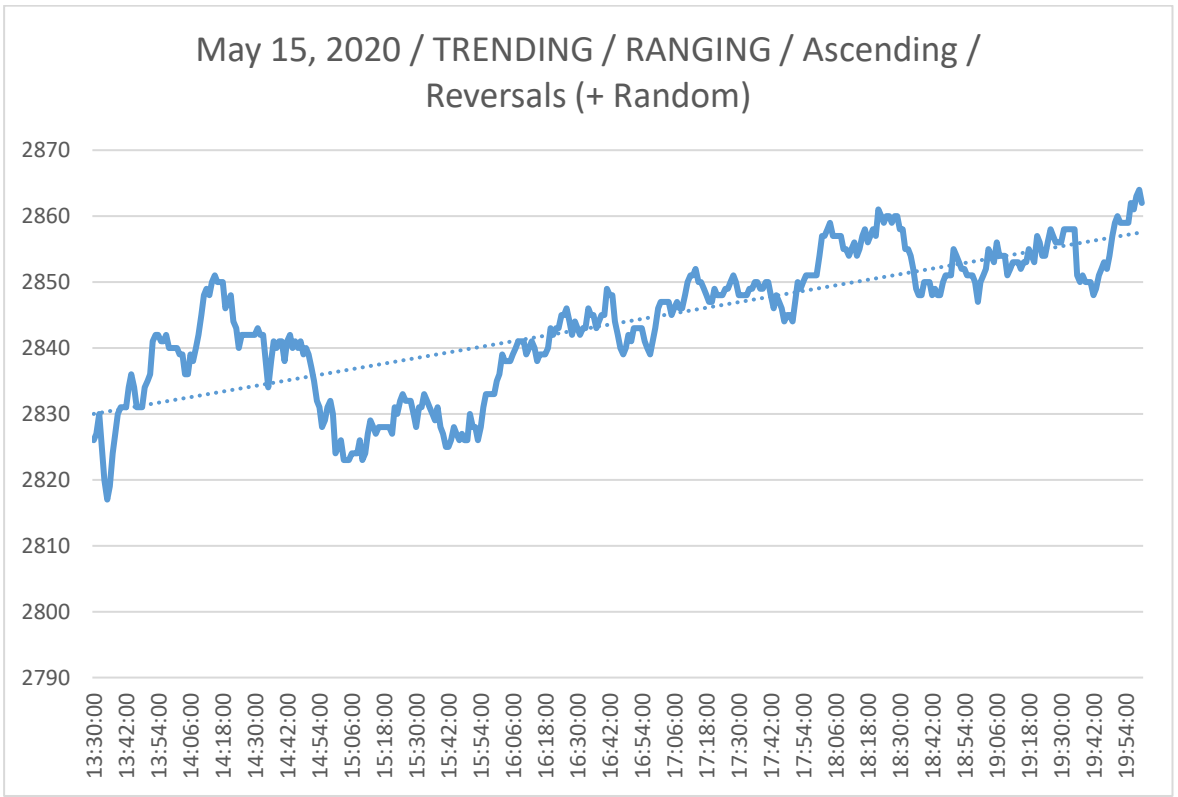

May 22, 2020 / TRENDING / RANGING / Ascending / Reversals
2960
2955
2950
2945
2940
2935
2930
2925
2920
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:30:00
14:42:00
14:54:00
15:06:00
15:18:00
15:30:00
15:42:00
15:54:00
16:06:00
16:18:00
16:30:00
16:42:00
16:54:00
17:06:00
17:18:00
17:30:00
17:42:00
17:54:00
18:06:00
18:18:00
18:30:00
18:42:00
18:54:00
19:06:00
19:18:00
19:30:00
19:42:00
19:54:00
20:06:00

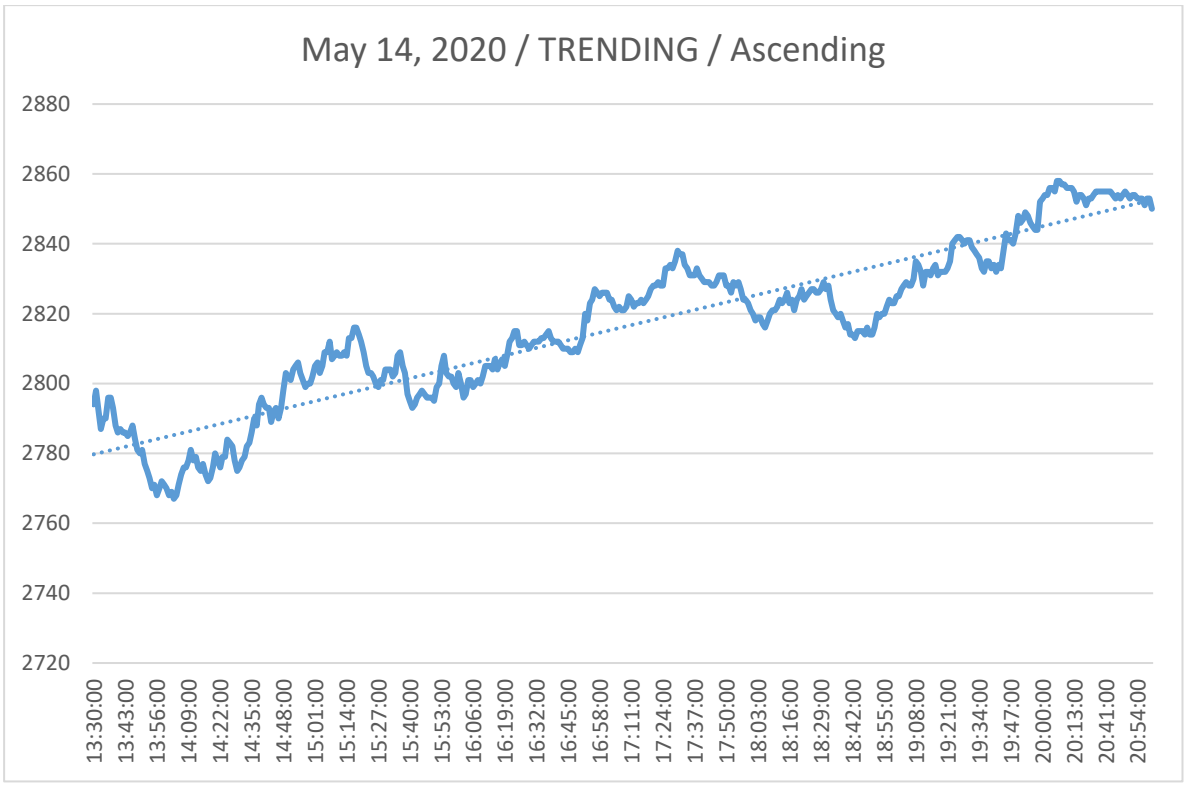

May 21, 2020 / TRENDING / Descending
2980
2970
2960
2950
2940
2930
2920
2910
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

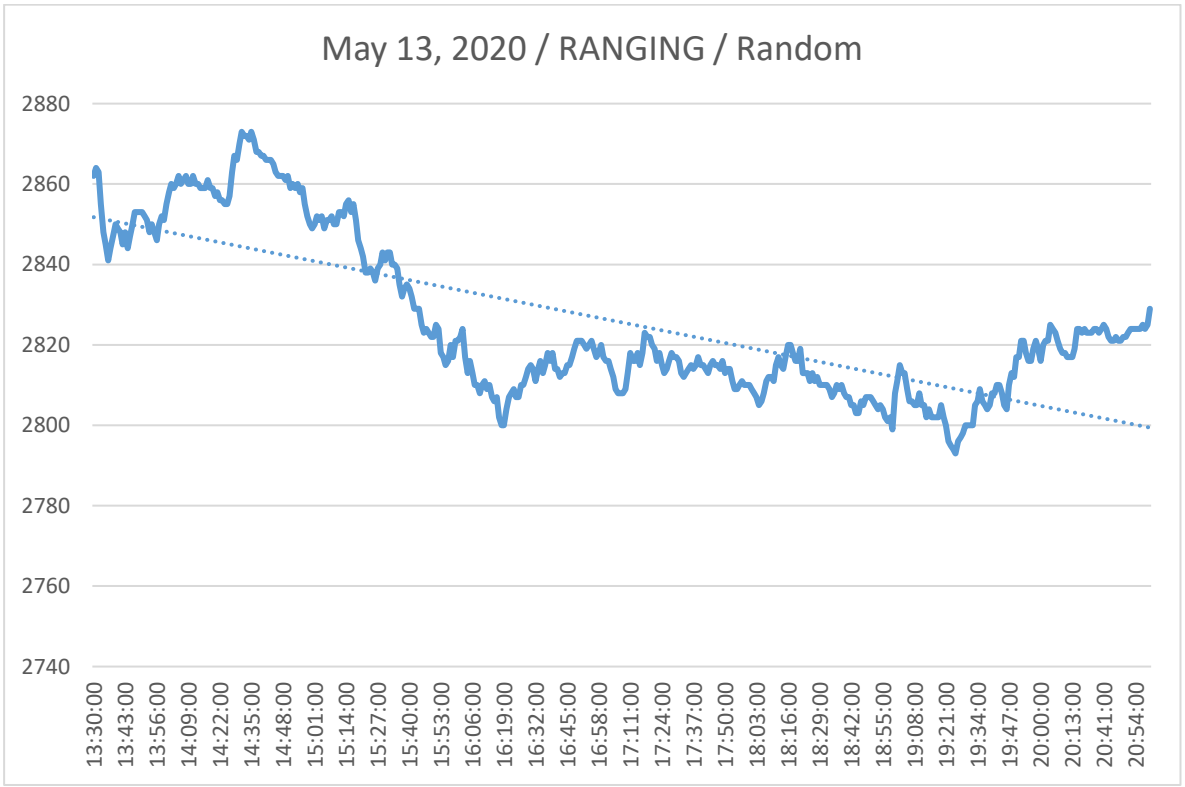

May 20, 2020 / RANGING / Reversals
2985
2980
2975
2970
2965
2960
2955
2950
2945
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

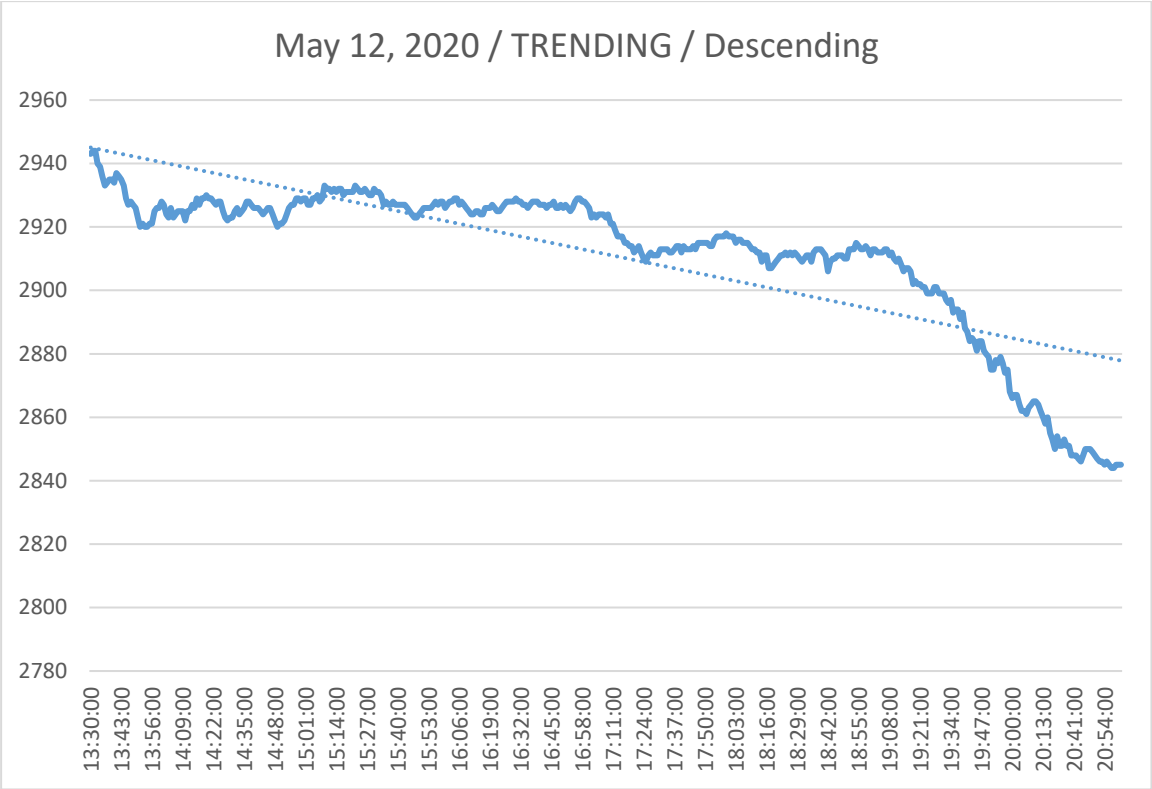

May 19, 2020 / RANGING / Random
2970
2960
2950
2940
2930
2920
2910
2900
2890
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

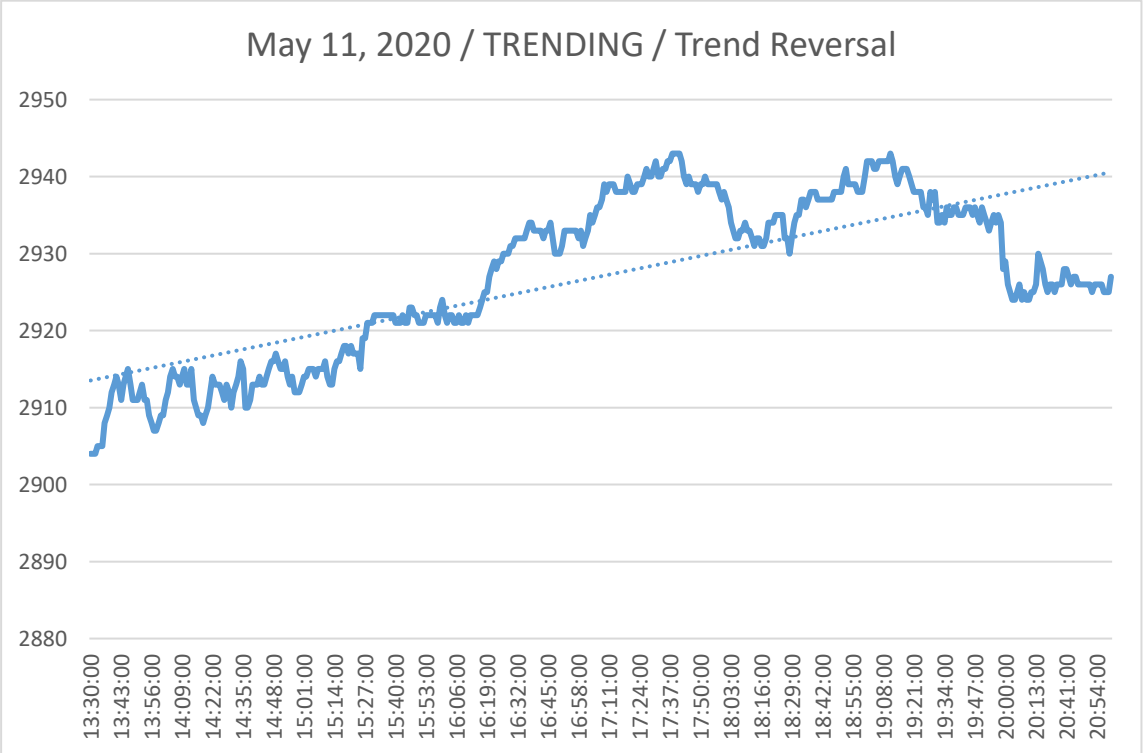

May 18, 2020 / TRENDING / Trend reversal
2980
2970
2960
2950
2940
2930
2920
2910
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

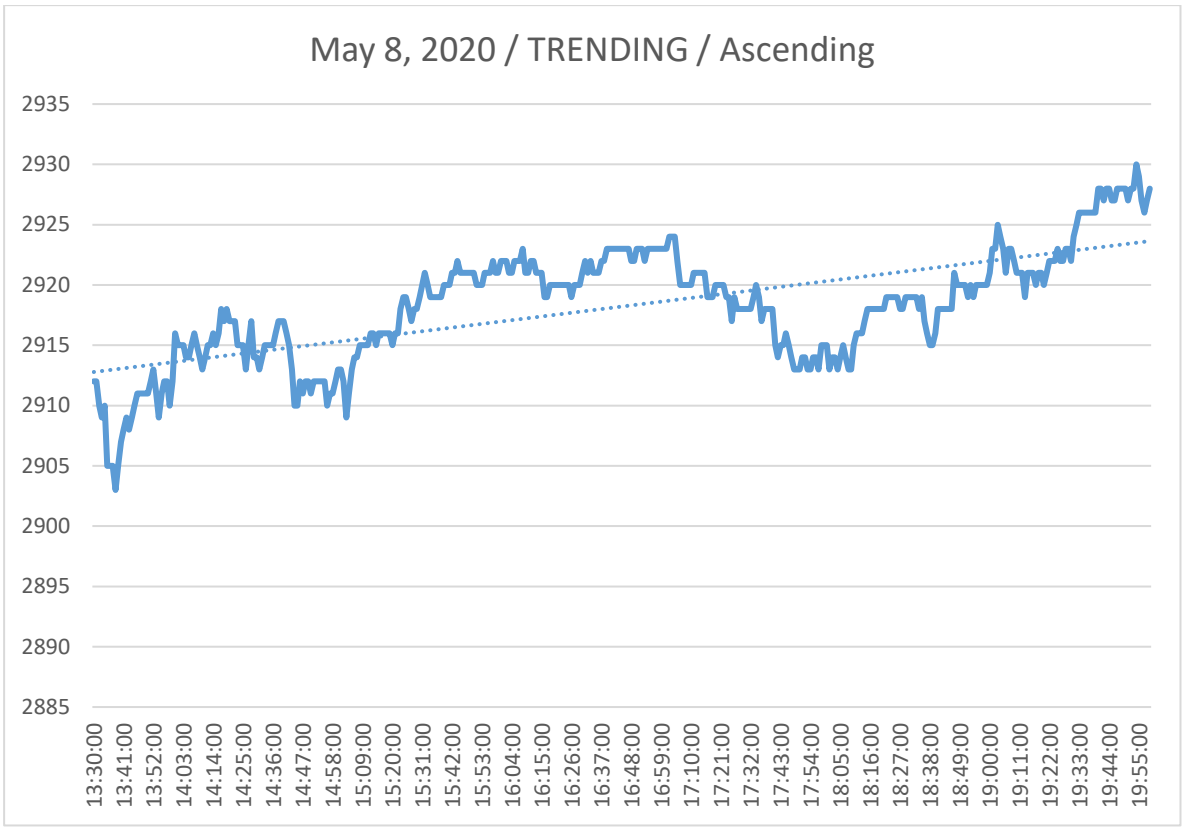

May 15, 2020 / TRENDING / RANGING / Ascending /
Reversals (+ Random)
2870
2860
2850
2840
2830
2820
2810
2800
2790
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:30:00
14:42:00
14:54:00
15:06:00
15:18:00
15:30:00
15:42:00
15:54:00
16:06:00
16:18:00
16:30:00
16:42:00
16:54:00
17:06:00
17:18:00
17:30:00
17:42:00
17:54:00
18:06:00
18:18:00
18:30:00
18:42:00
18:54:00
19:06:00
19:18:00
19:30:00
19:42:00
19:54:00

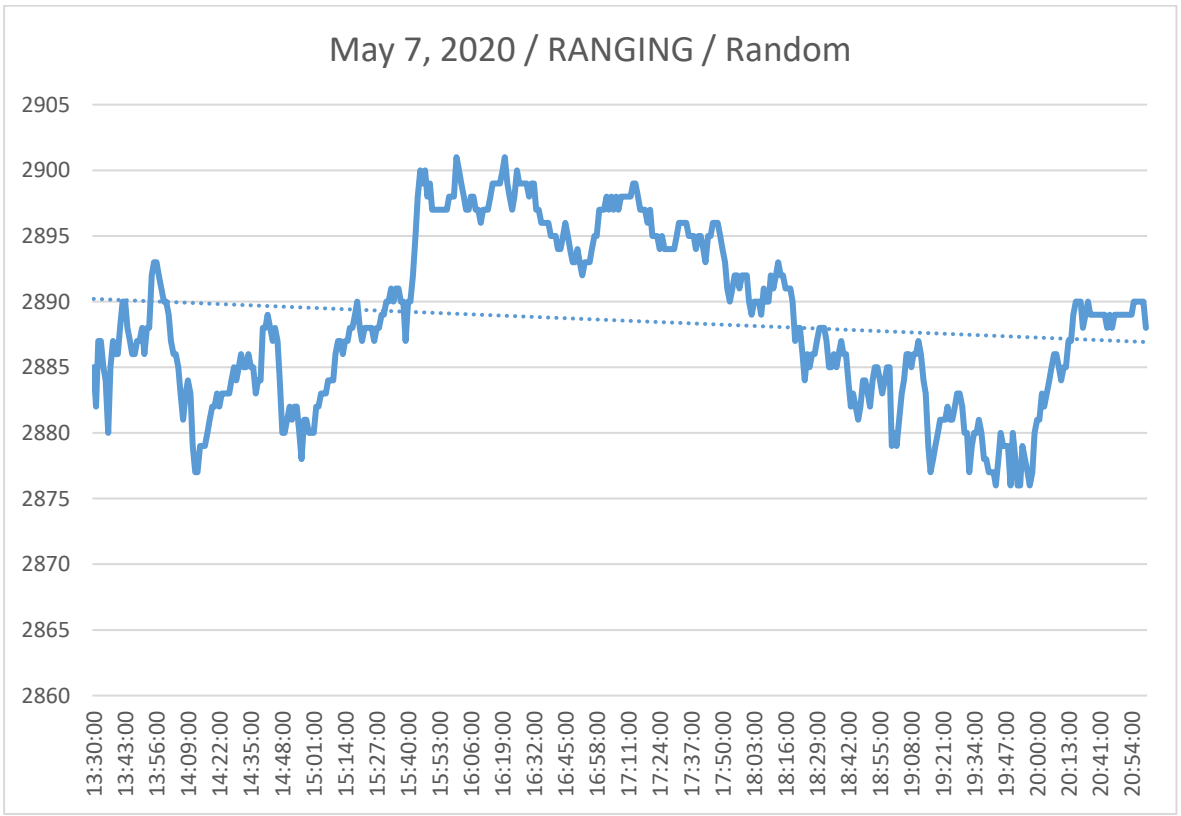

May 14, 2020 / TRENDING / Ascending
2880
2860
2840
2820
2800
2780
2760
2740
2720
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

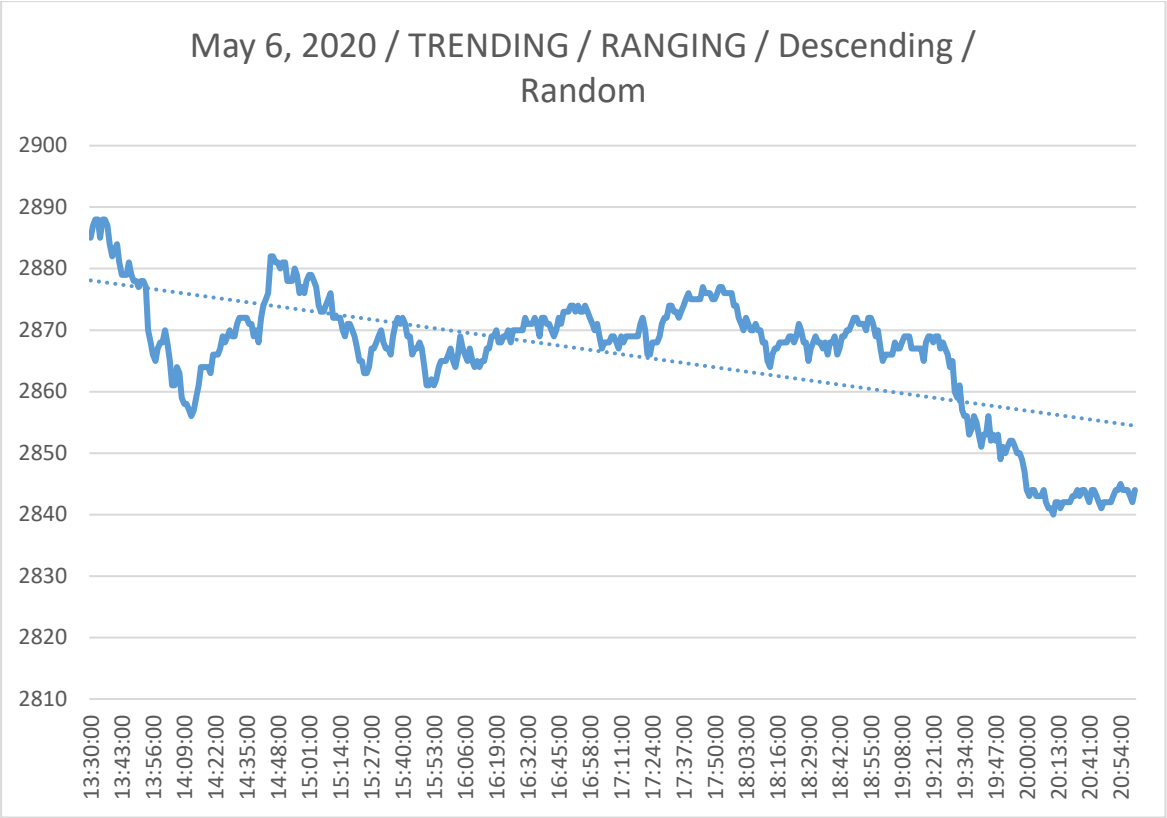

May 13, 2020 / RANGING / Random
2880
2860
2840
2820
2800
2780
2760
2740
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

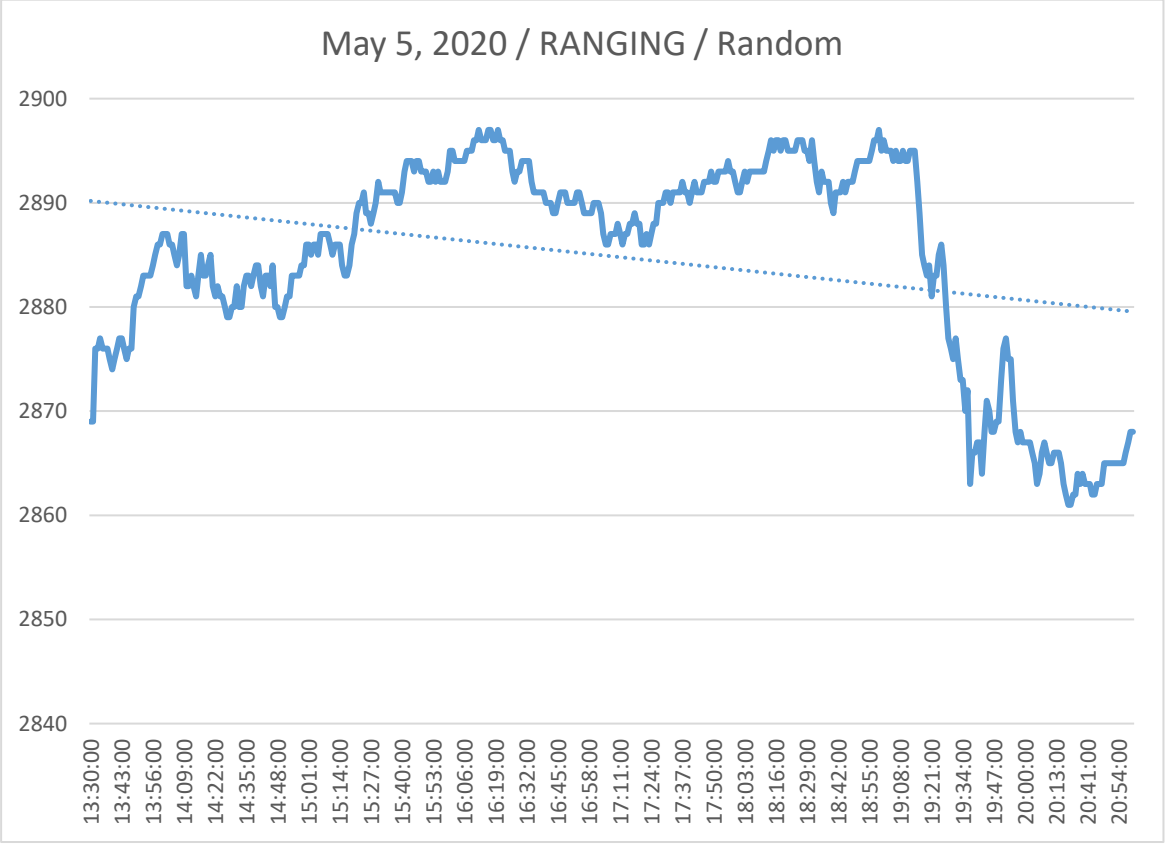

May 12, 2020 / TRENDING / Descending
2960
2940
2920
2900
2880
2860
2840
2820
2800
2780
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

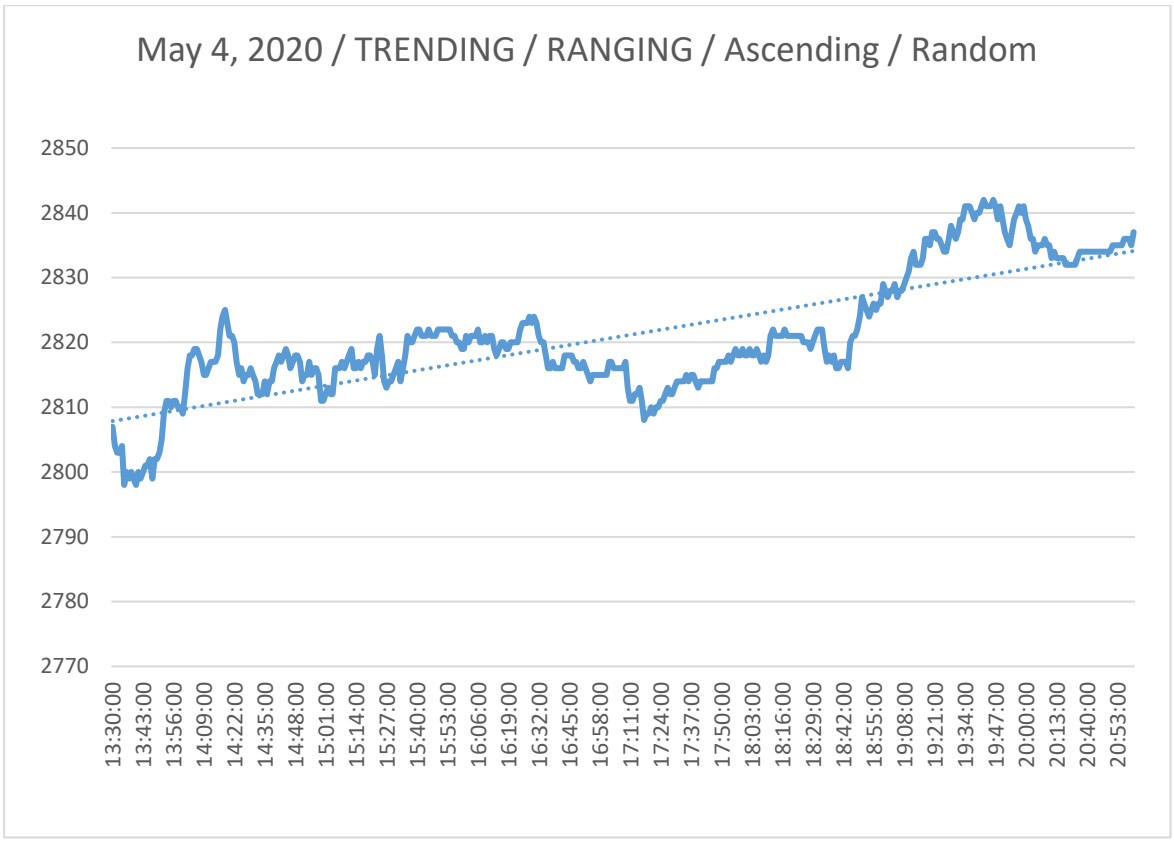

May 11, 2020 / TRENDING / Trend Reversal
2950
2940
2930
2920
2910
2900
2890
2880
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

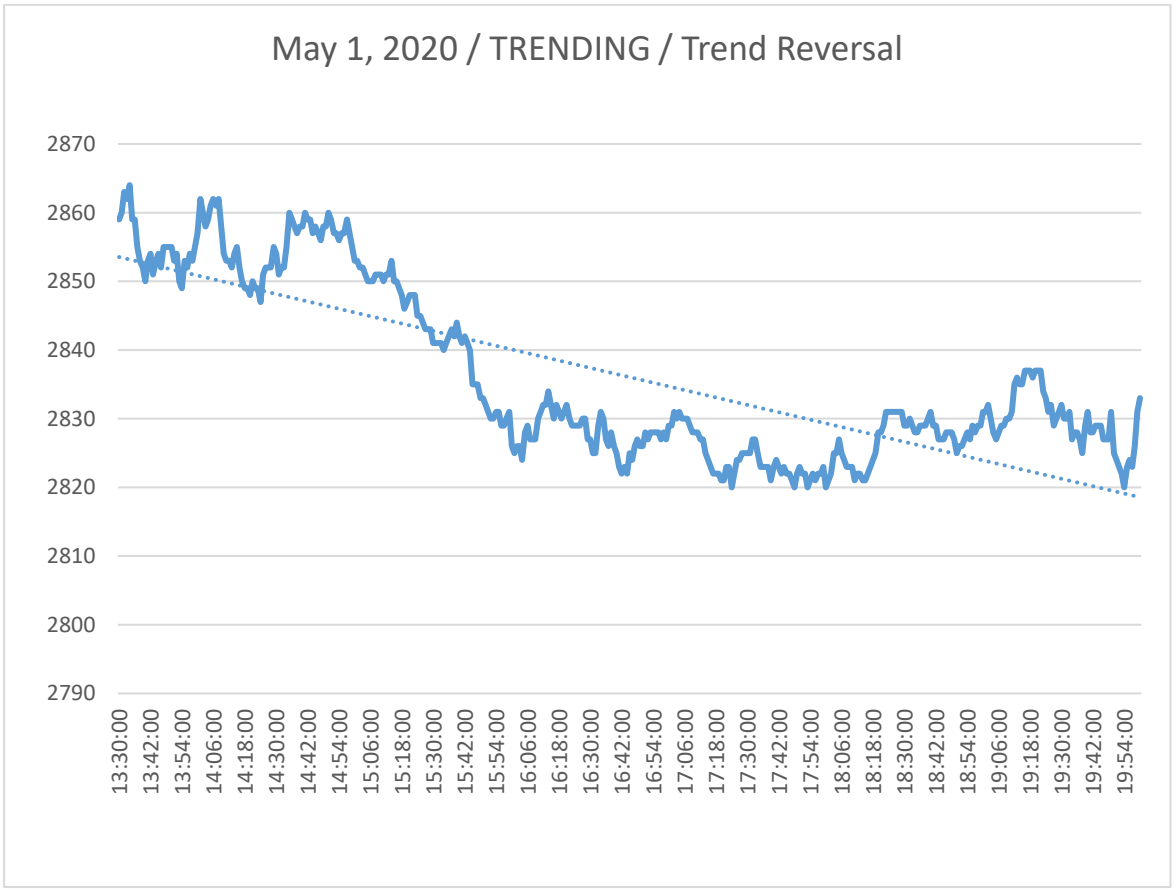

May 8, 2020 / TRENDING / Ascending
2935
2930
2925
2920
2915
2910
2905
2900
2895
2890
2885
13:30:00
13:41:00
13:52:00
14:03:00
14:14:00
14:25:00
14:36:00
14:47:00
14:58:00
15:09:00
15:20:00
15:31:00
15:42:00
15:53:00
16:04:00
16:15:00
16:26:00
16:37:00
16:48:00
16:59:00
17:10:00
17:21:00
17:32:00
17:43:00
17:54:00
18:05:00
18:16:00
18:27:00
18:38:00
18:49:00
19:00:00
19:11:00
19:22:00
19:33:00
19:44:00
19:55:00

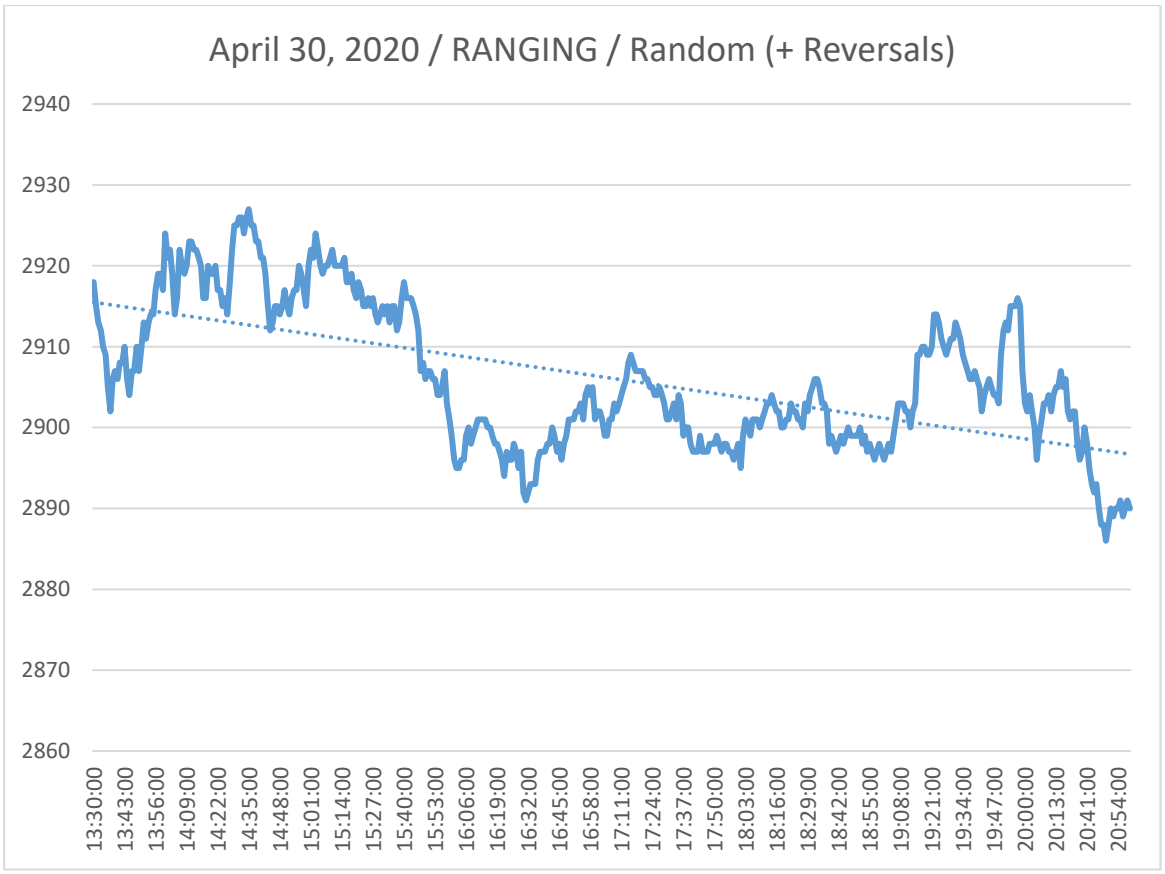

May 7, 2020 / RANGING / Random
2905
2900
2895
2890
2885
2880
2875
2870
2865
2860
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

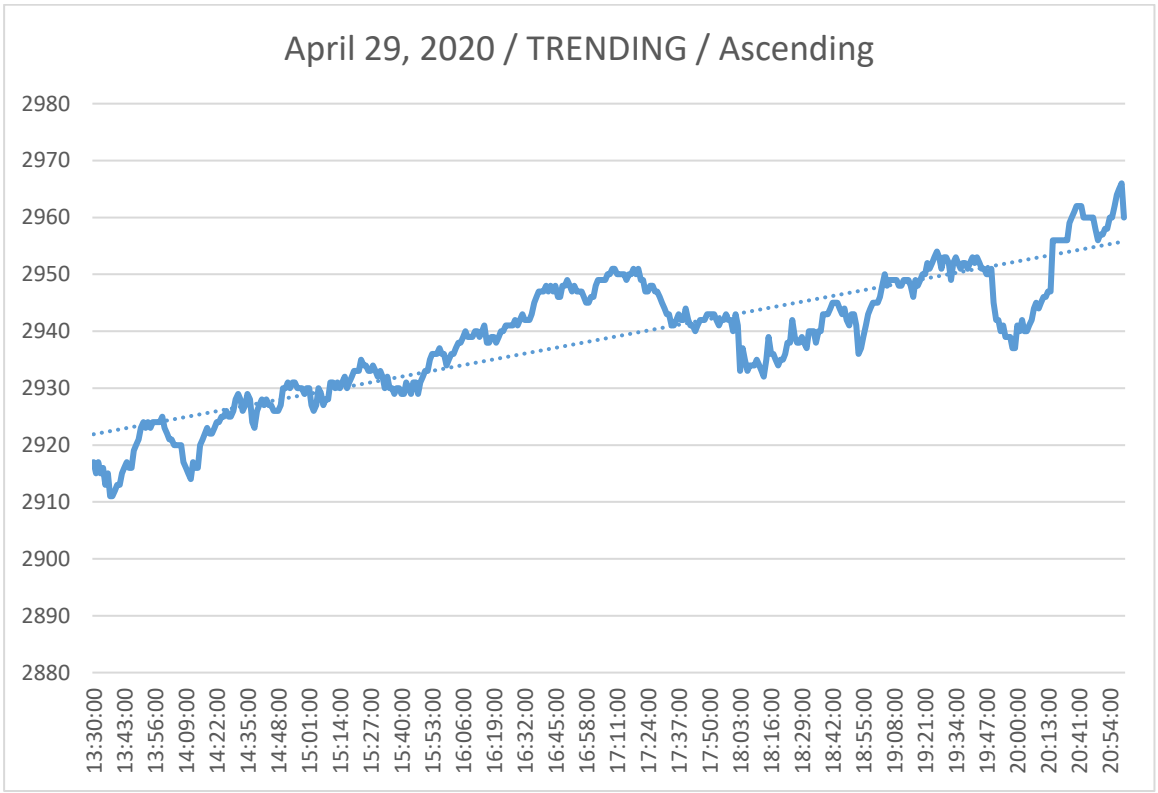

May 6, 2020 / TRENDING / RANGING / Descending / Random
2900
2890
2880
2870
2860
2850
2840
2830
2820
2810
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

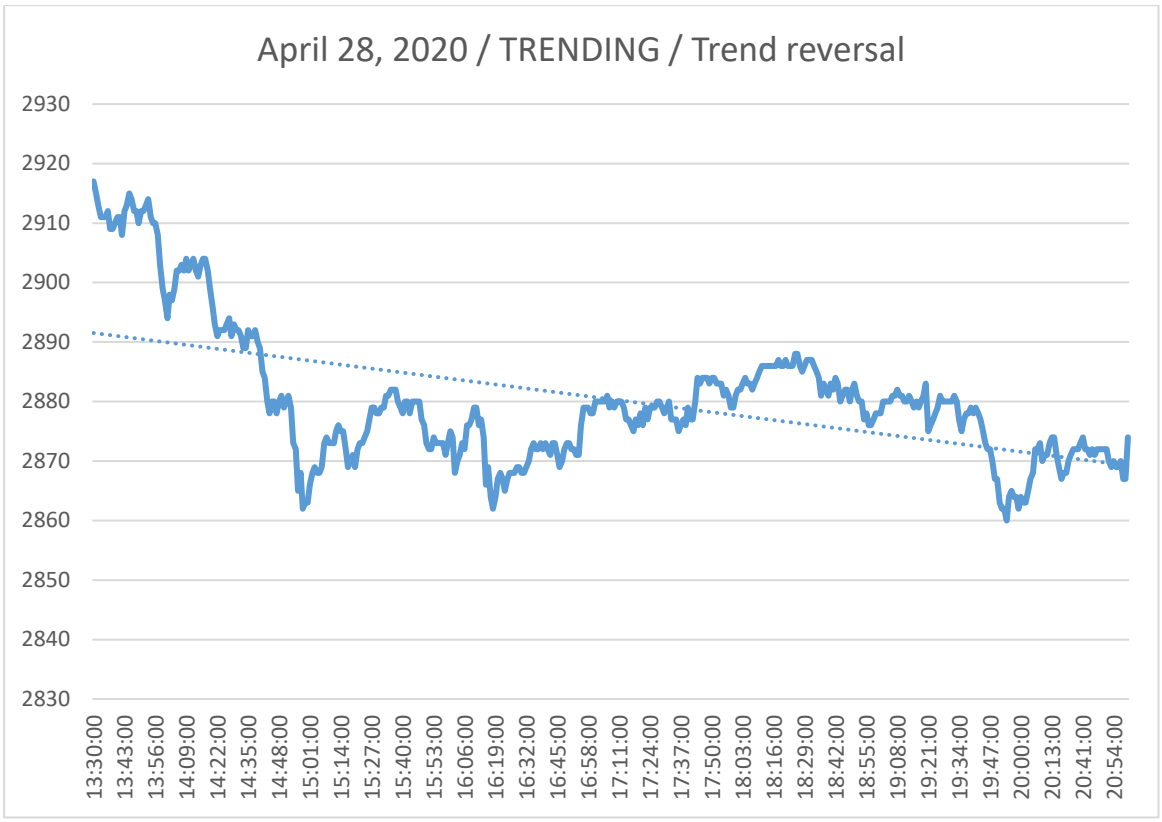

May 5, 2020 / RANGING / Random
2900
2890
2880
2870
2860
2850
2840
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

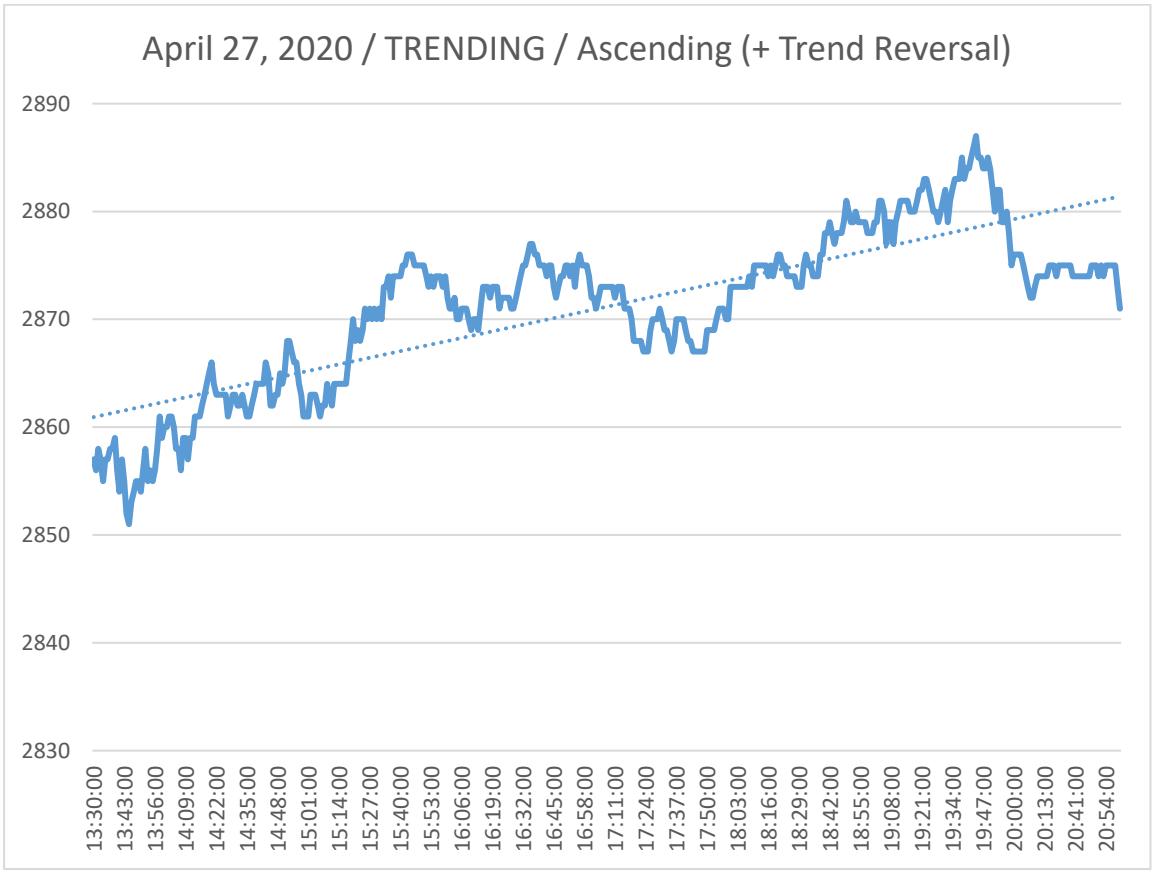

May 4, 2020 / TRENDING / RANGING / Ascending / Random
2850
2840
2830
2820
2810
2800
2790
2780
2770
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:40:00
20:53:00

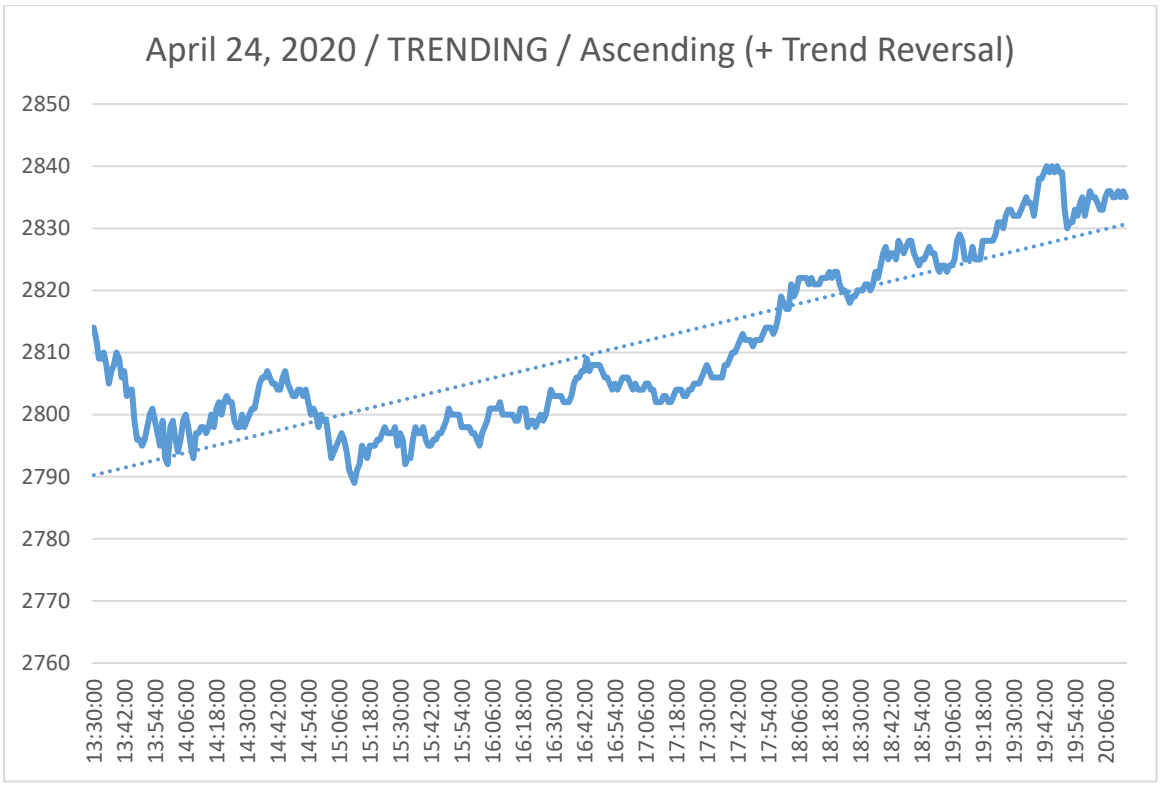

May 1, 2020 / TRENDING / Trend Reversal
2870
2860
2850
2840
2830
2820
2810
2800
2790
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:30:00
14:42:00
14:54:00
15:06:00
15:18:00
15:30:00
15:42:00
15:54:00
16:06:00
16:18:00
16:30:00
16:42:00
16:54:00
17:06:00
17:18:00
17:30:00
17:42:00
17:54:00
18:06:00
18:18:00
18:30:00
18:42:00
18:54:00
19:06:00
19:18:00
19:30:00
19:42:00
19:54:00

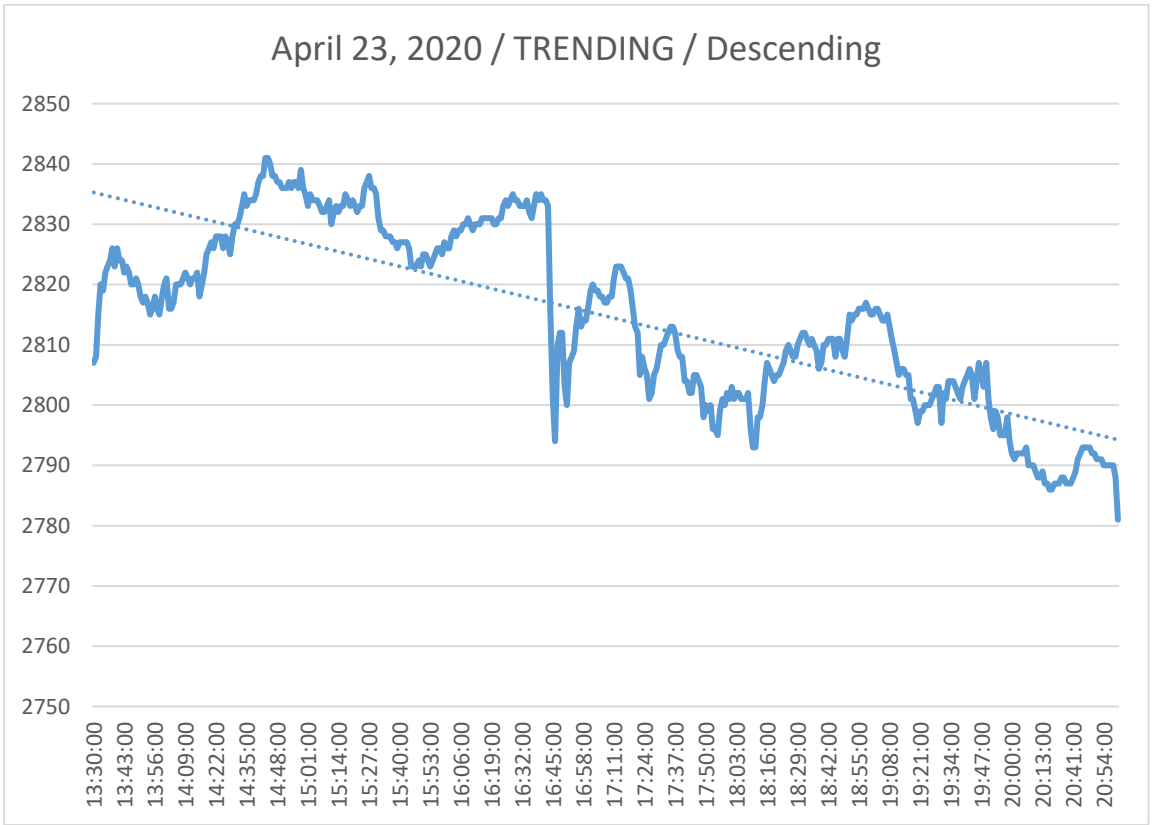

April 30, 2020 / RANGING / Random (+ Reversals)
2940
2930
2920
2910
2900
2890
2880
2870
2860
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

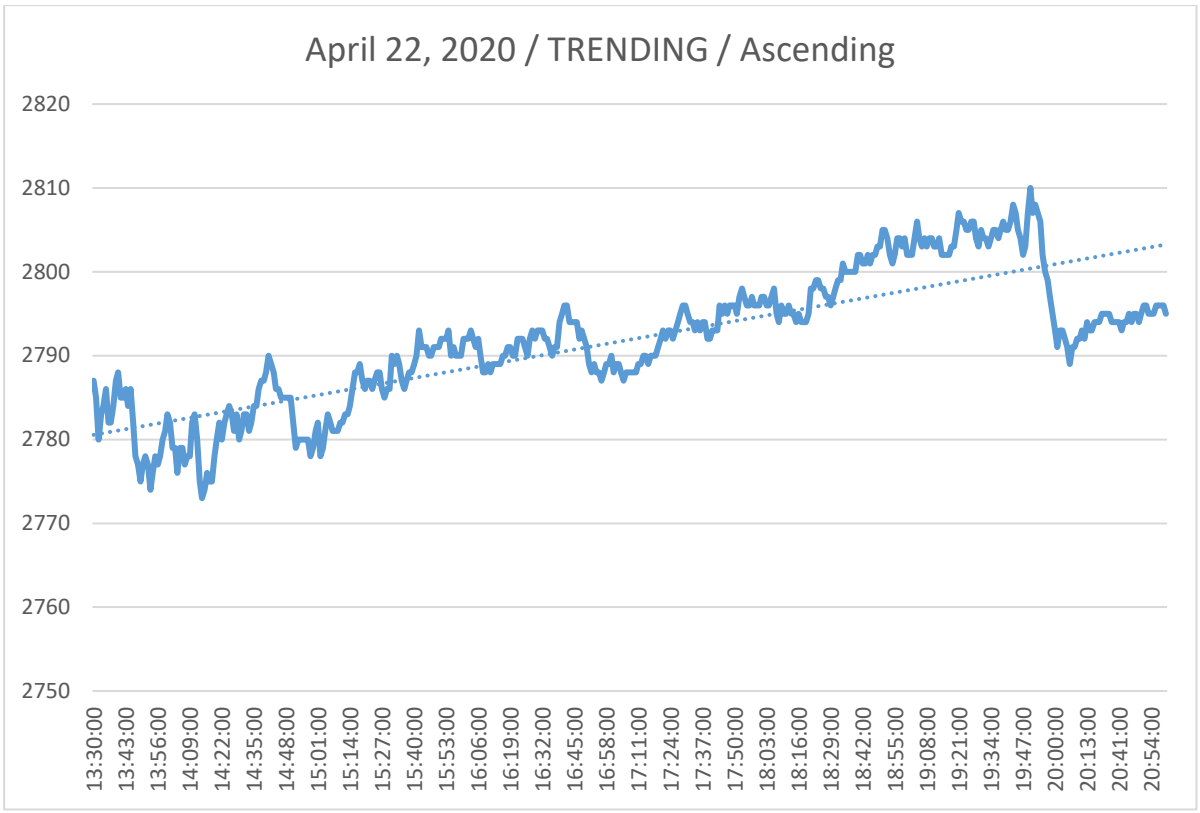

April 29, 2020 / TRENDING / Ascending
2980
2970
2960
2950
2940
2930
2920
2910
2900
2890
2880
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

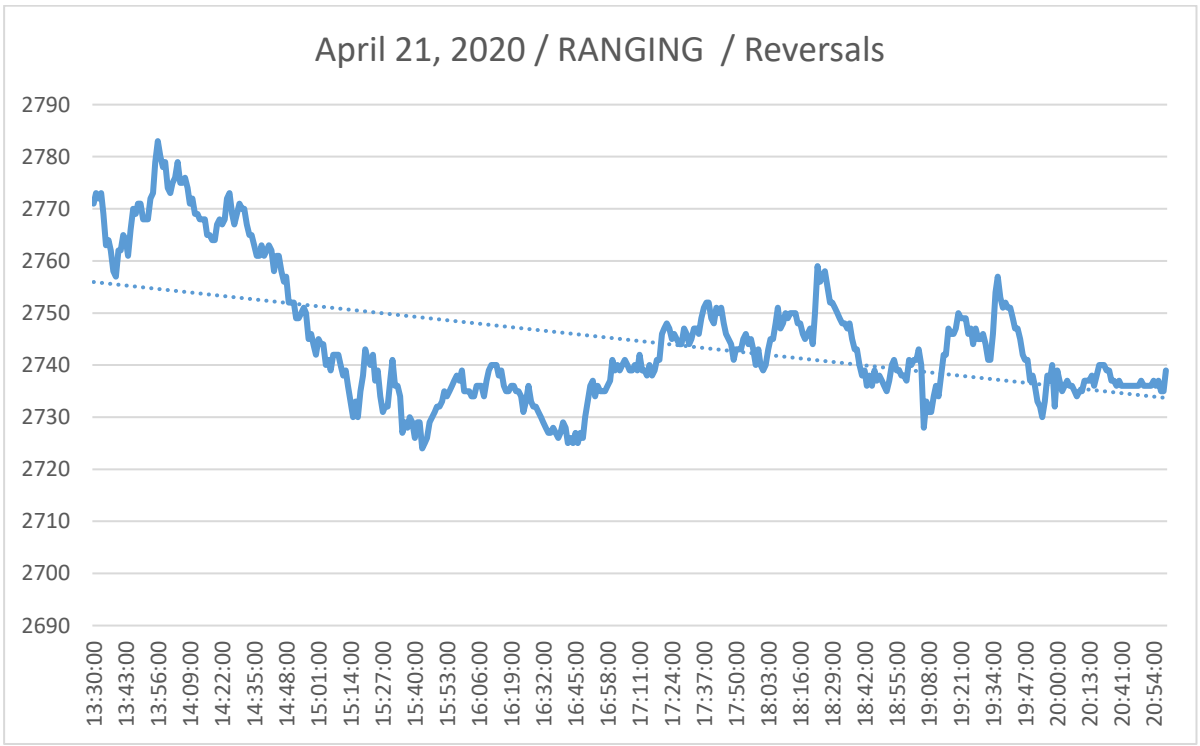

April 28, 2020 / TRENDING / Trend reversal
2930
2920
2910
2900
2890
2880
2870
2860
2850
2840
2830
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

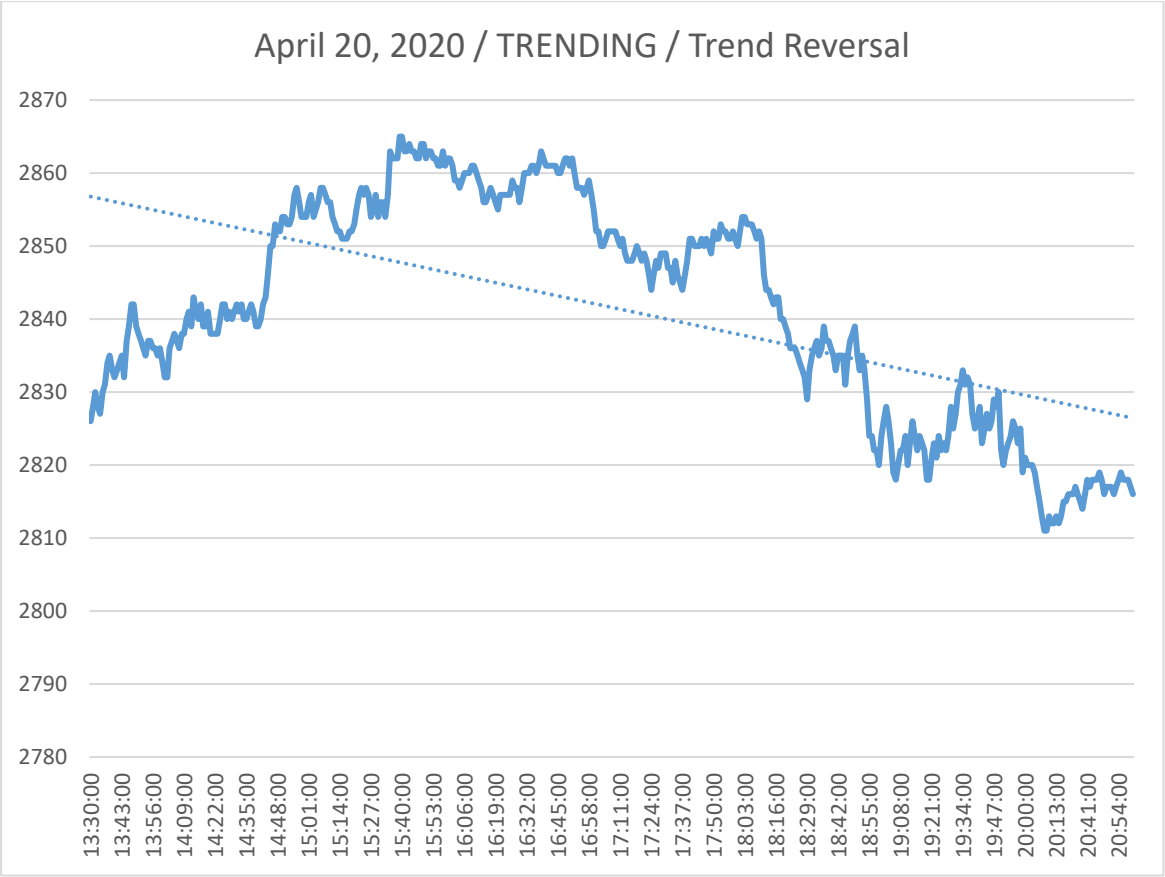

April 27, 2020 / TRENDING / Ascending (+ Trend Reversal)
2890
2880
2870
2860
2850
2840
2830
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

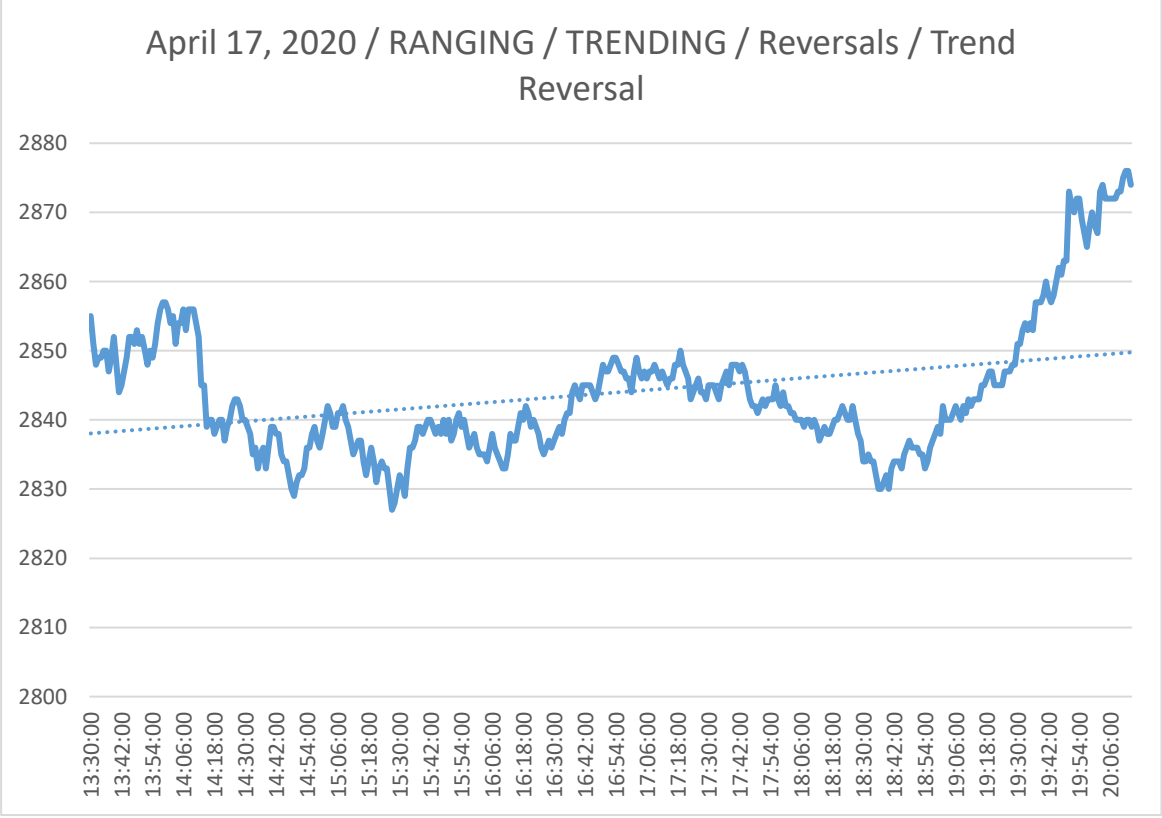

April 24, 2020 / TRENDING / Ascending (+ Trend Reversal)
2850
2840
2830
2820
2810
2800
2790
2780
2770
2760
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:30:00
14:42:00
14:54:00
15:06:00
15:18:00
15:30:00
15:42:00
15:54:00
16:06:00
16:18:00
16:30:00
16:42:00
16:54:00
17:06:00
17:18:00
17:30:00
17:42:00
17:54:00
18:06:00
18:18:00
18:30:00
18:42:00
18:54:00
19:06:00
19:18:00
19:30:00
19:42:00
19:54:00
20:06:00

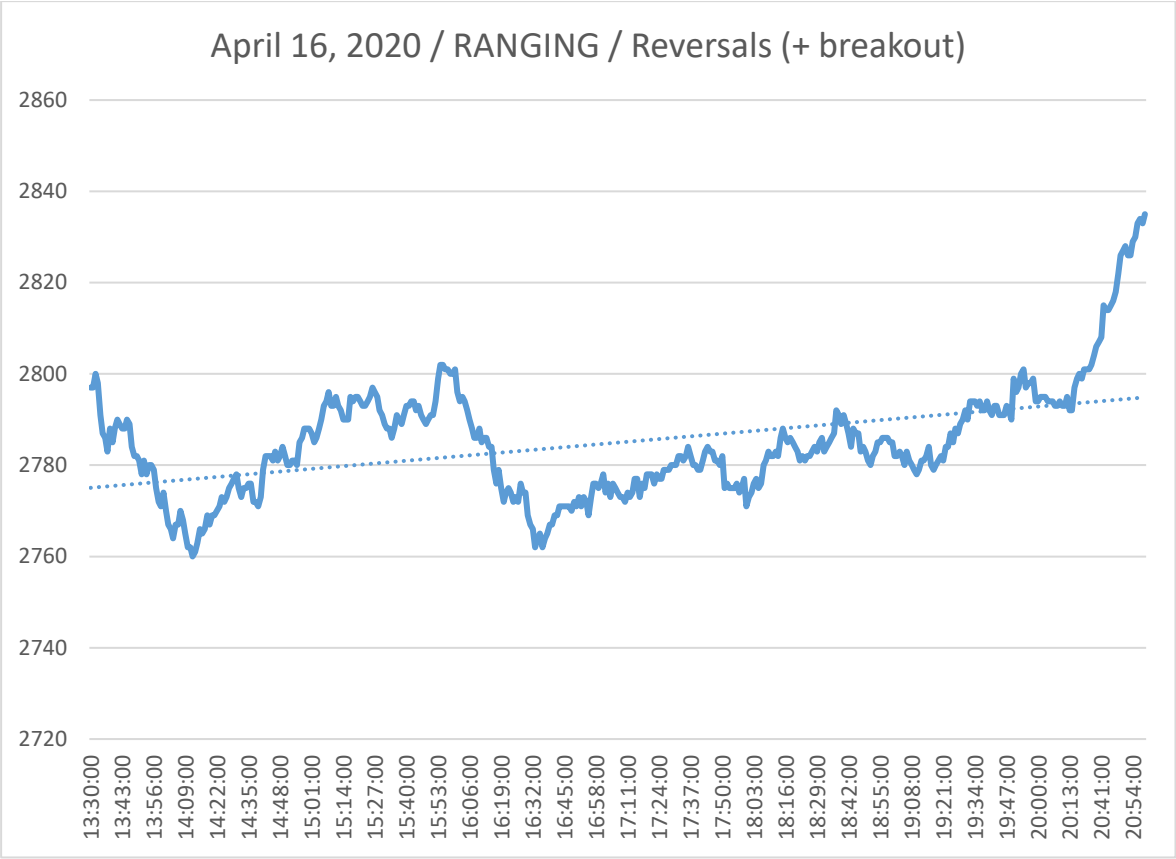

April 23, 2020 / TRENDING / Descending
2850
2840
2830
2820
2810
2800
2790
2780
2770
2760
2750
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

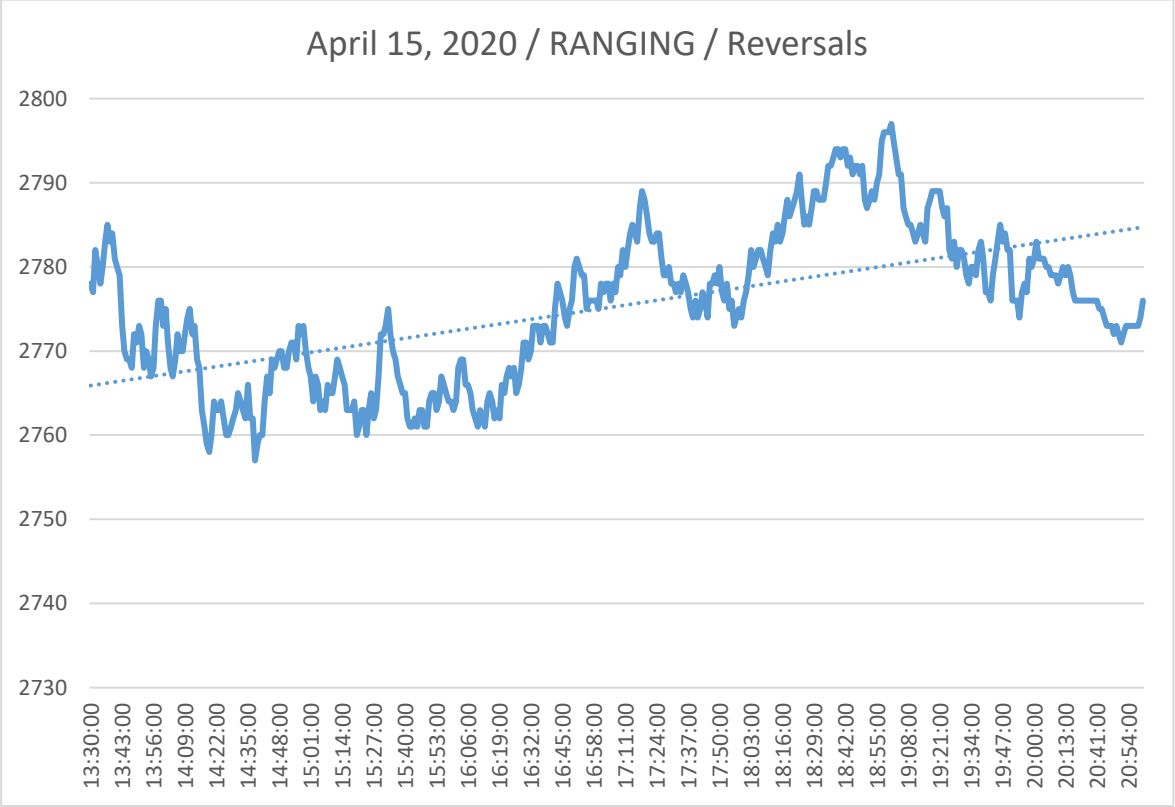

April 22, 2020 / TRENDING / Ascending
2820
2810
2800
2790
2780
2770
2760
2750
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

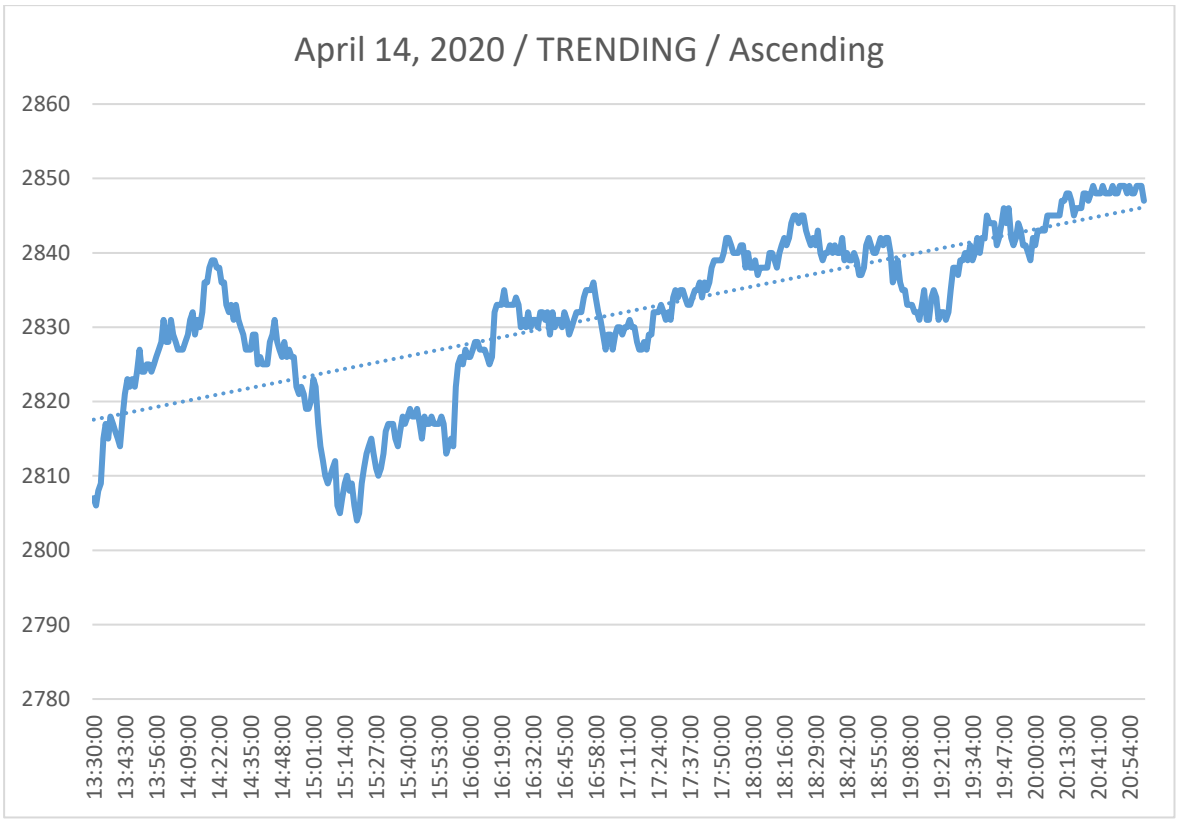

April 21, 2020 / RANGING / Reversals
2790
2780
2770
2760
2750
2740
2730
2720
2710
2700
2690
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

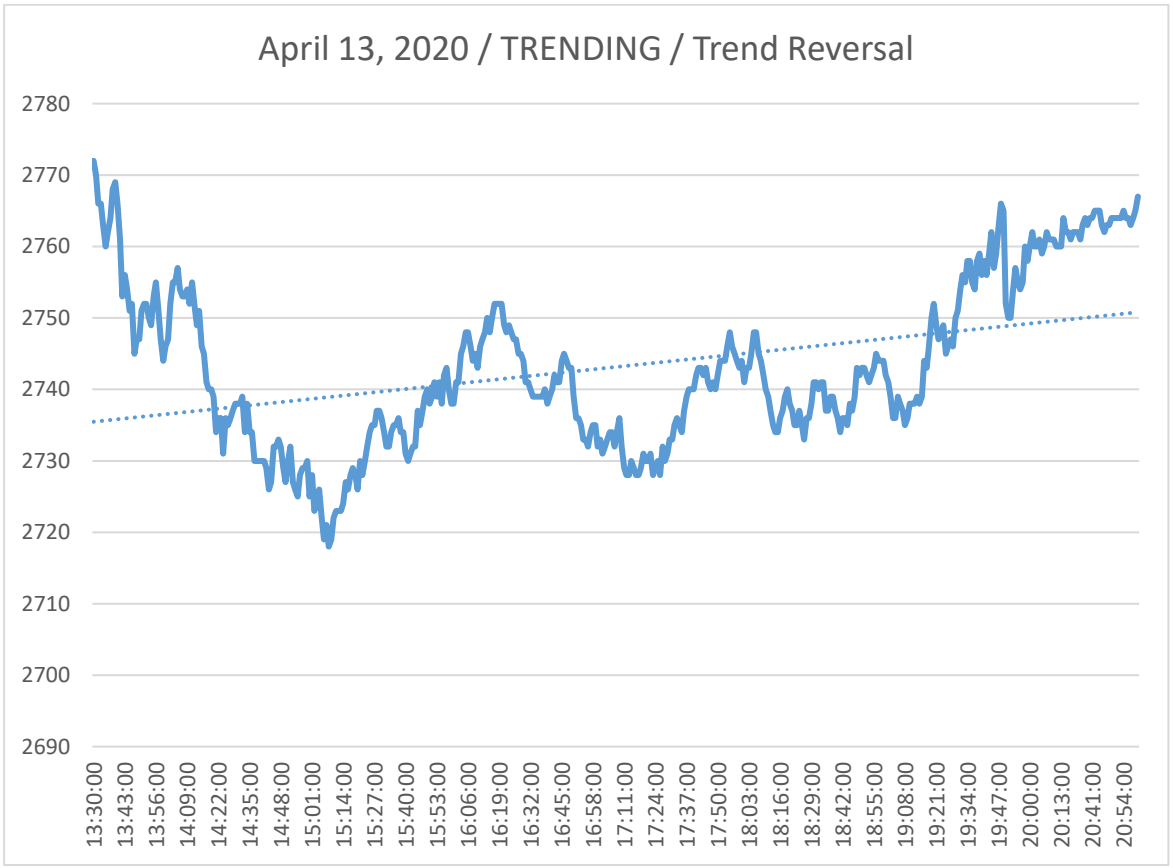

April 20, 2020 / TRENDING / Trend Reversal
2870
2860
2850
2840
2830
2820
2810
2800
2790
2780
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

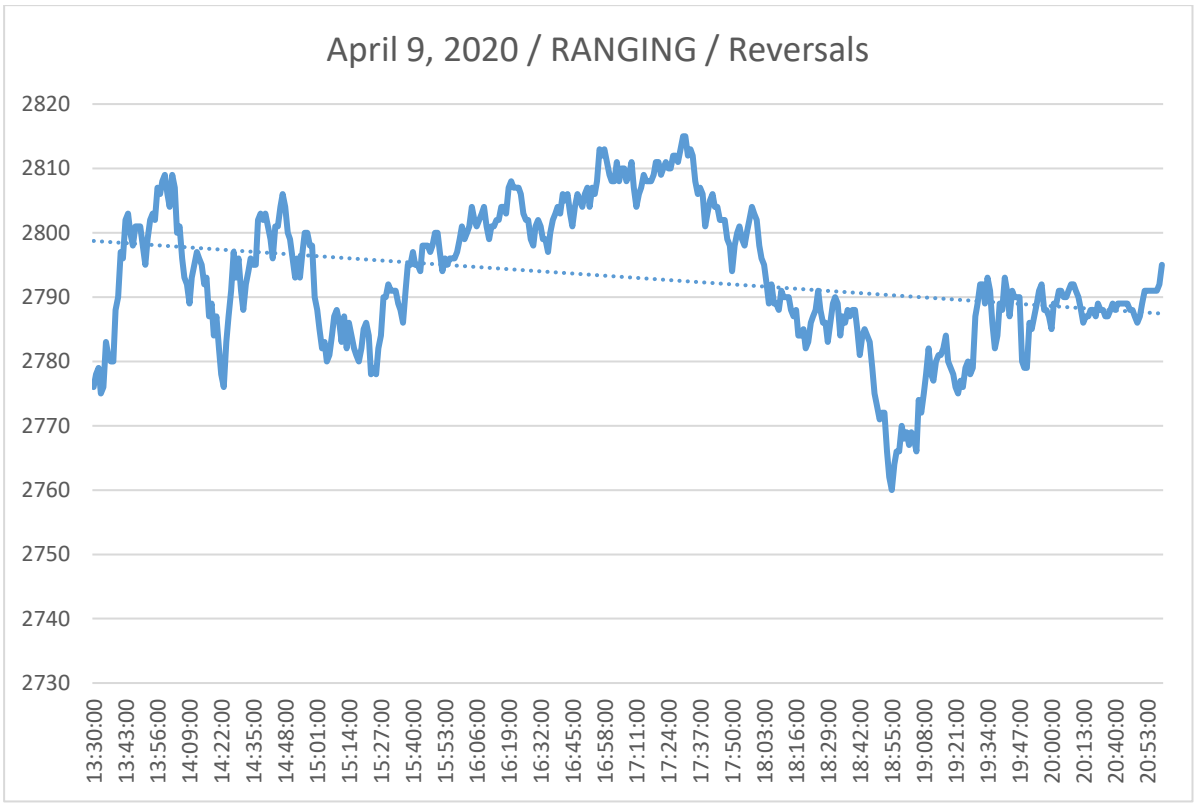

April 17, 2020 / RANGING / TRENDING / Reversals / Trend Reversal
2880
2870
2860
2850
2840
2830
2820
2810
2800
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:30:00
14:42:00
14:54:00
15:06:00
15:18:00
15:30:00
15:42:00
15:54:00
16:06:00
16:18:00
16:30:00
16:42:00
16:54:00
17:06:00
17:18:00
17:30:00
17:42:00
17:54:00
18:06:00
18:18:00
18:30:00
18:42:00
18:54:00
19:06:00
19:18:00
19:30:00
19:42:00
19:54:00
20:06:00

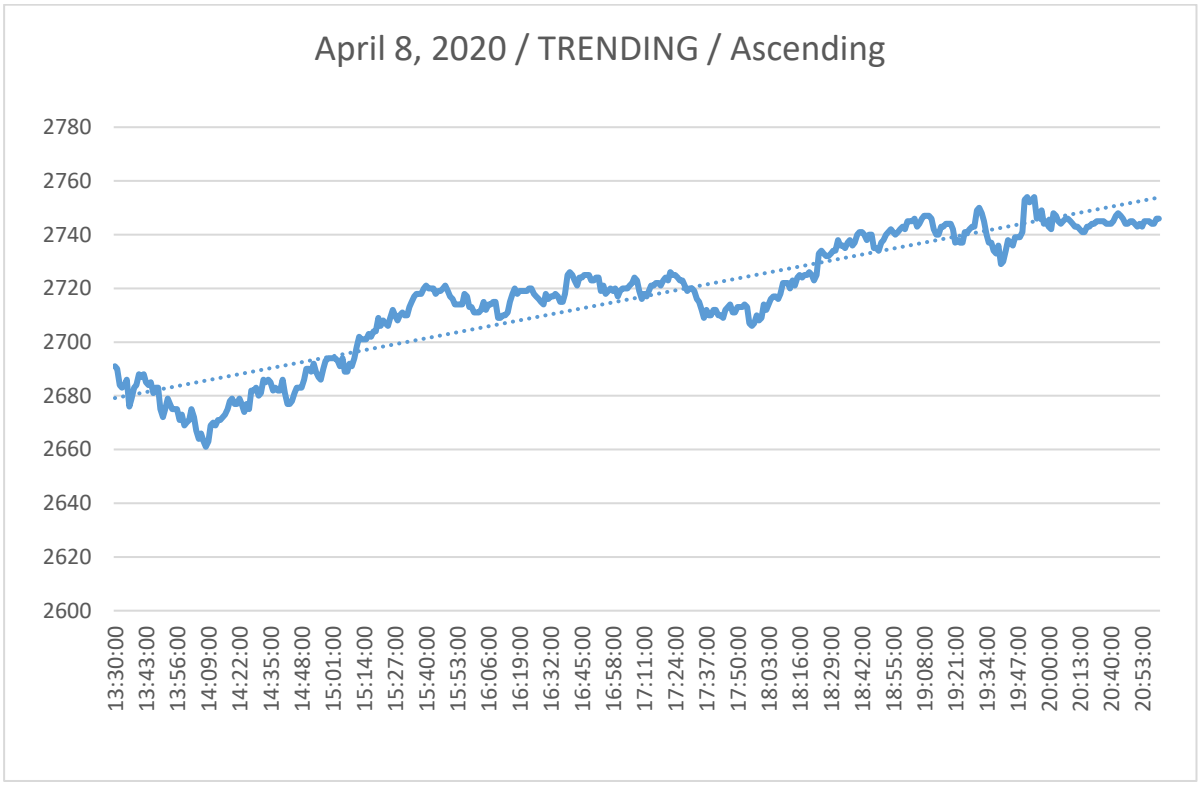

April 16, 2020 / RANGING / Reversals (+ breakout)
2860
2840
2820
2800
2780
2760
2740
2720
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

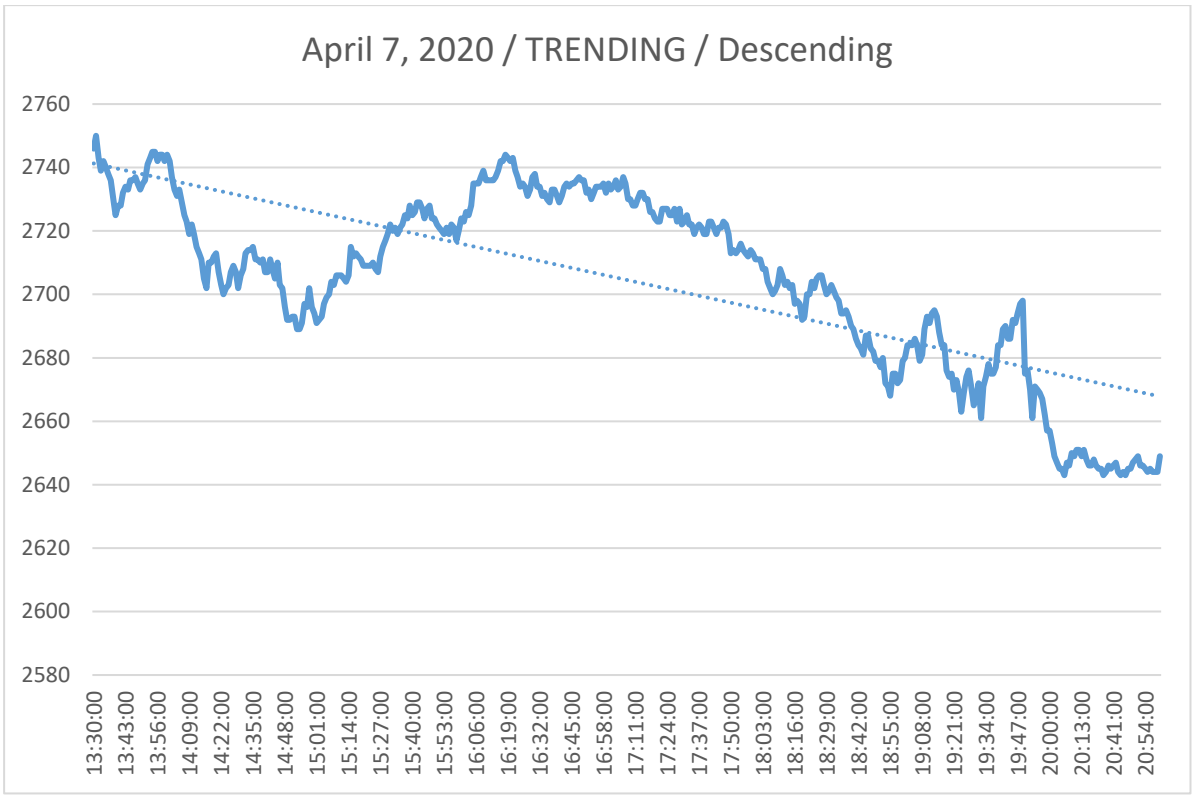

April 15, 2020 / RANGING / Reversals
2800
2790
2780
2770
2760
2750
2740
2730
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

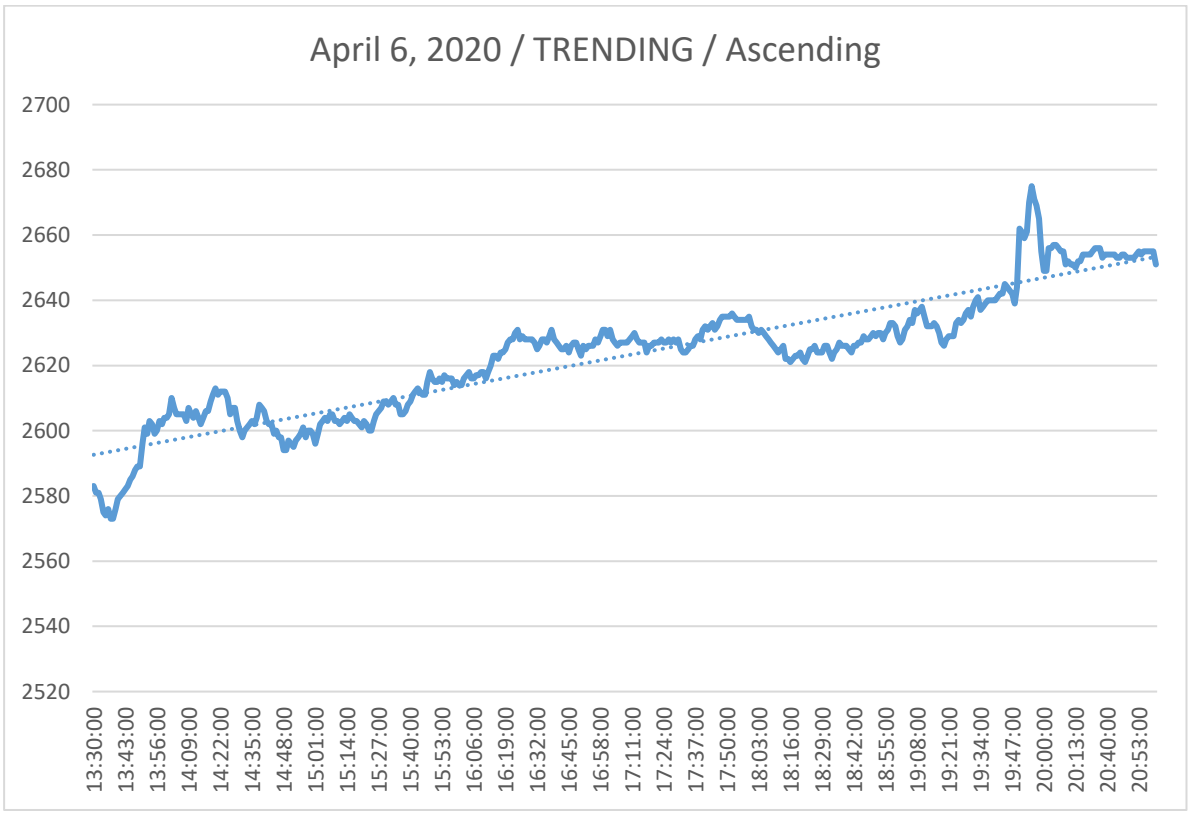

April 14, 2020 / TRENDING / Ascending
2860
2850
2840
2830
2820
2810
2800
2790
2780
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

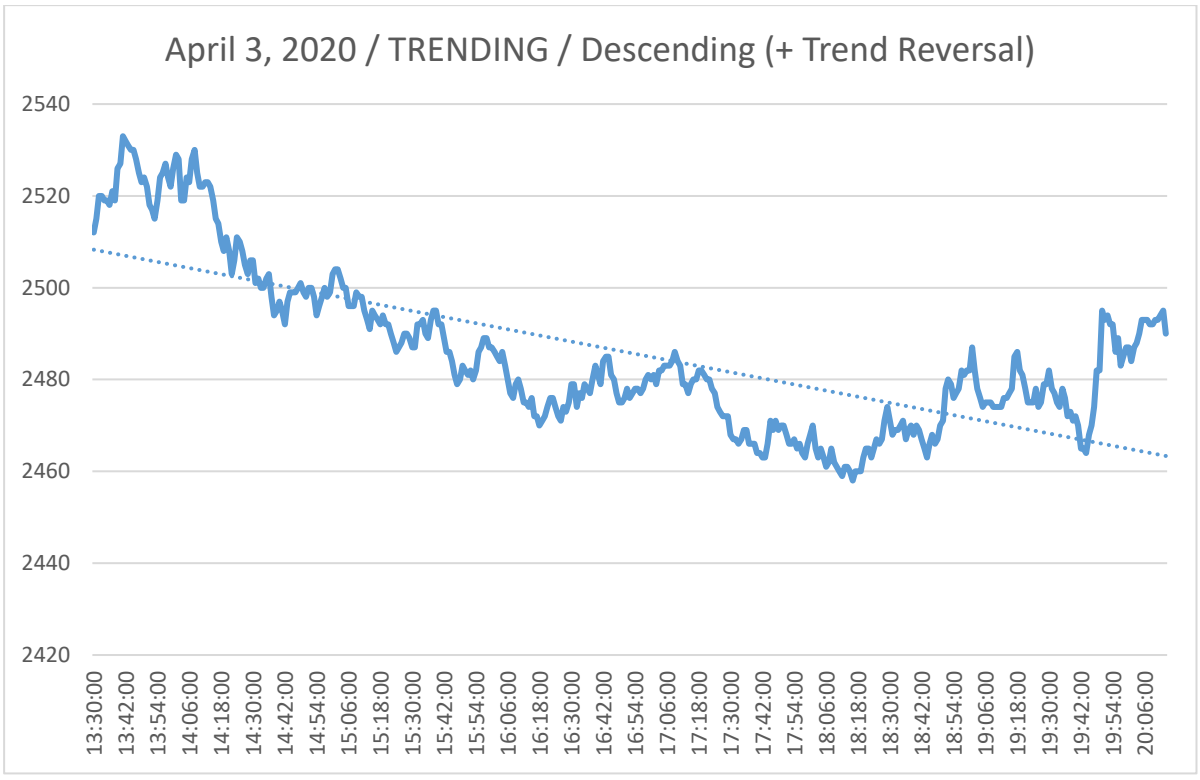

April 13, 2020 / TRENDING / Trend Reversal
2780
2770
2760
2750
2740
2730
2720
2710
2700
2690
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

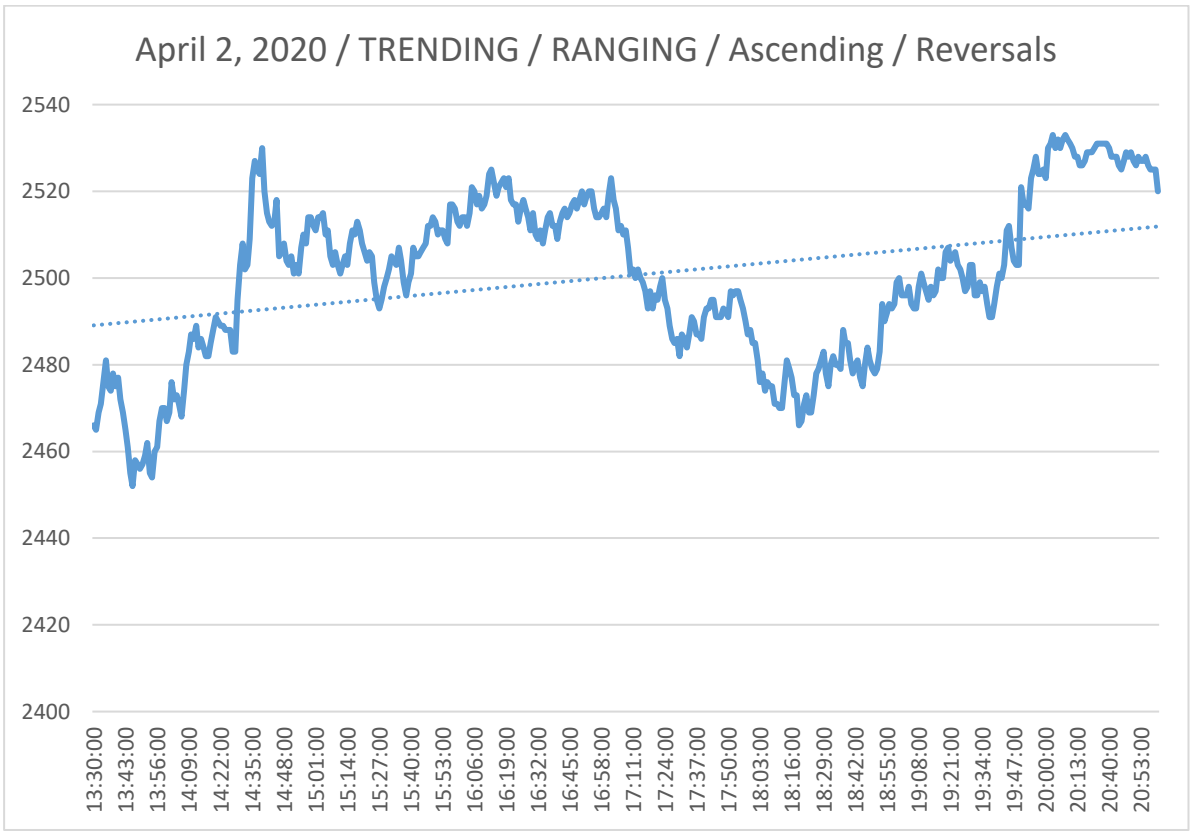

April 9, 2020 / RANGING / Reversals
2820
2810
2800
2790
2780
2770
2760
2750
2740
2730
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:40:00
20:53:00

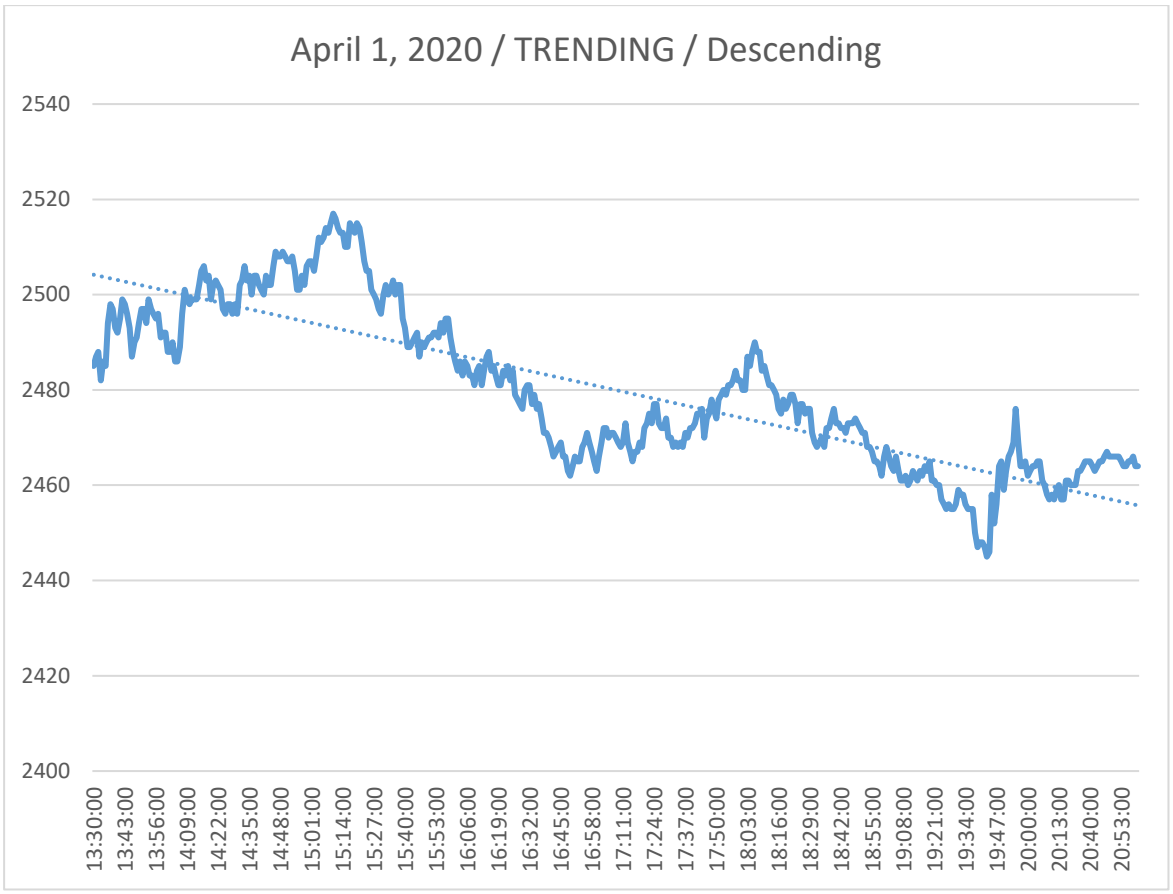

April 8, 2020 / TRENDING / Ascending
2780
2760
2740
2720
2700
2680
2660
2640
2620
2600
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:40:00
20:53:00

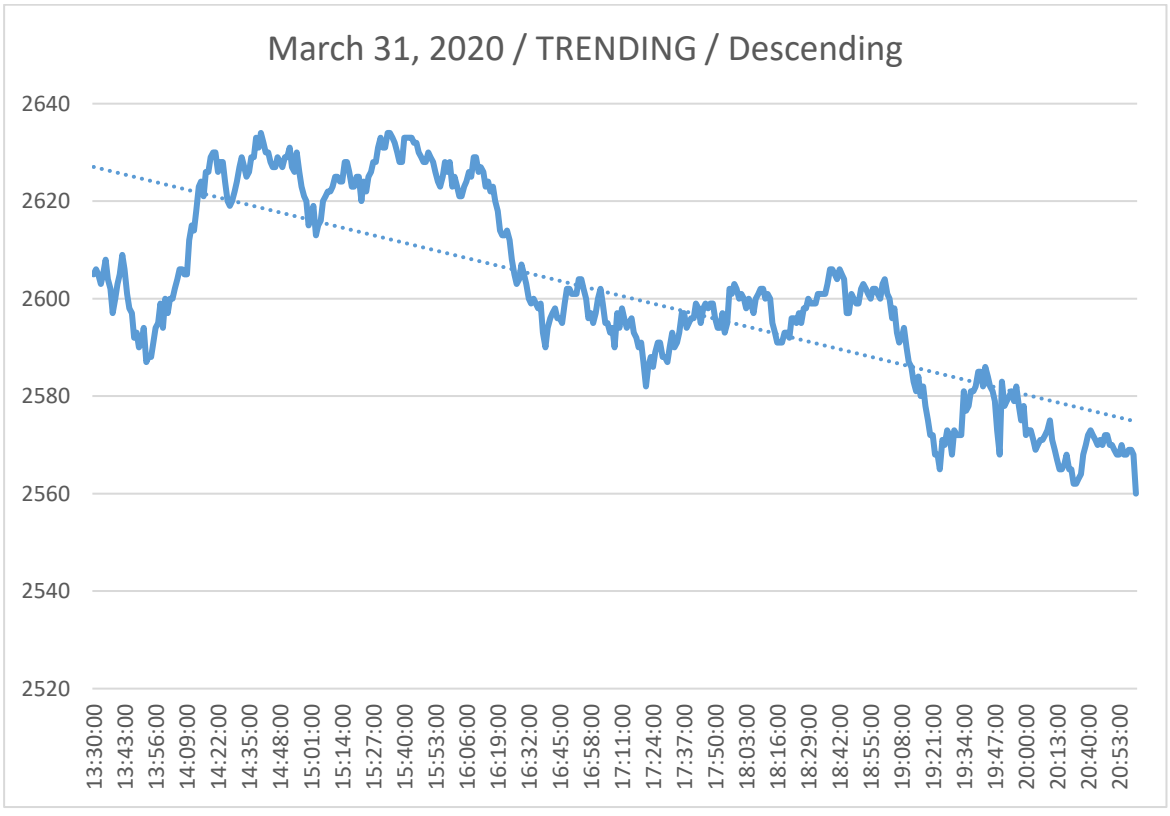

April 7, 2020 / TRENDING / Descending
2760
2740
2720
2700
2680
2660
2640
2620
2600
2580
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

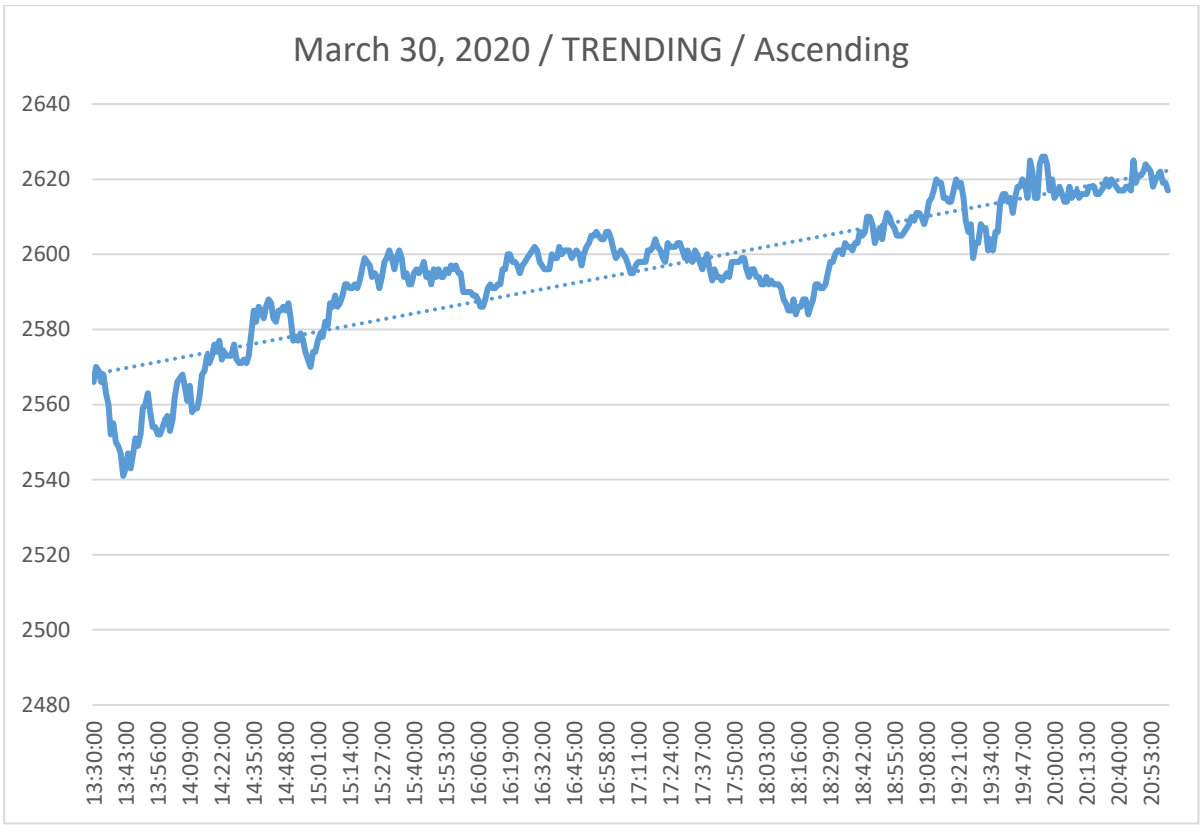

April 6, 2020 / TRENDING / Ascending
2700
2680
2660
2640
2620
2600
2580
2560
2540
2520
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:40:00
20:53:00

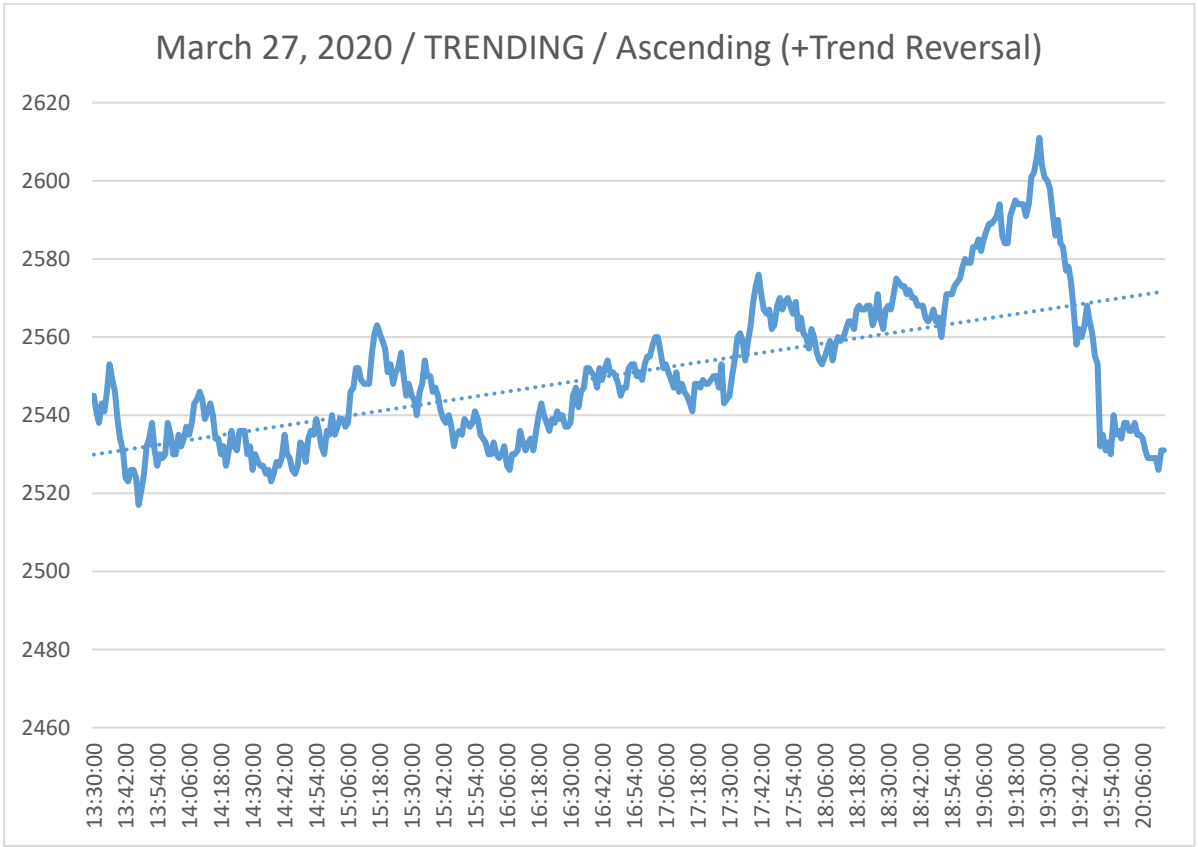

April 3, 2020 / TRENDING / Descending (+ Trend Reversal)
2540
2520
2500
2480
2460
2440
2420
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:30:00
14:42:00
14:54:00
15:06:00
15:18:00
15:30:00
15:42:00
15:54:00
16:06:00
16:18:00
16:30:00
16:42:00
16:54:00
17:06:00
17:18:00
17:30:00
17:42:00
17:54:00
18:06:00
18:18:00
18:30:00
18:42:00
18:54:00
19:06:00
19:18:00
19:30:00
19:42:00
19:54:00
20:06:00

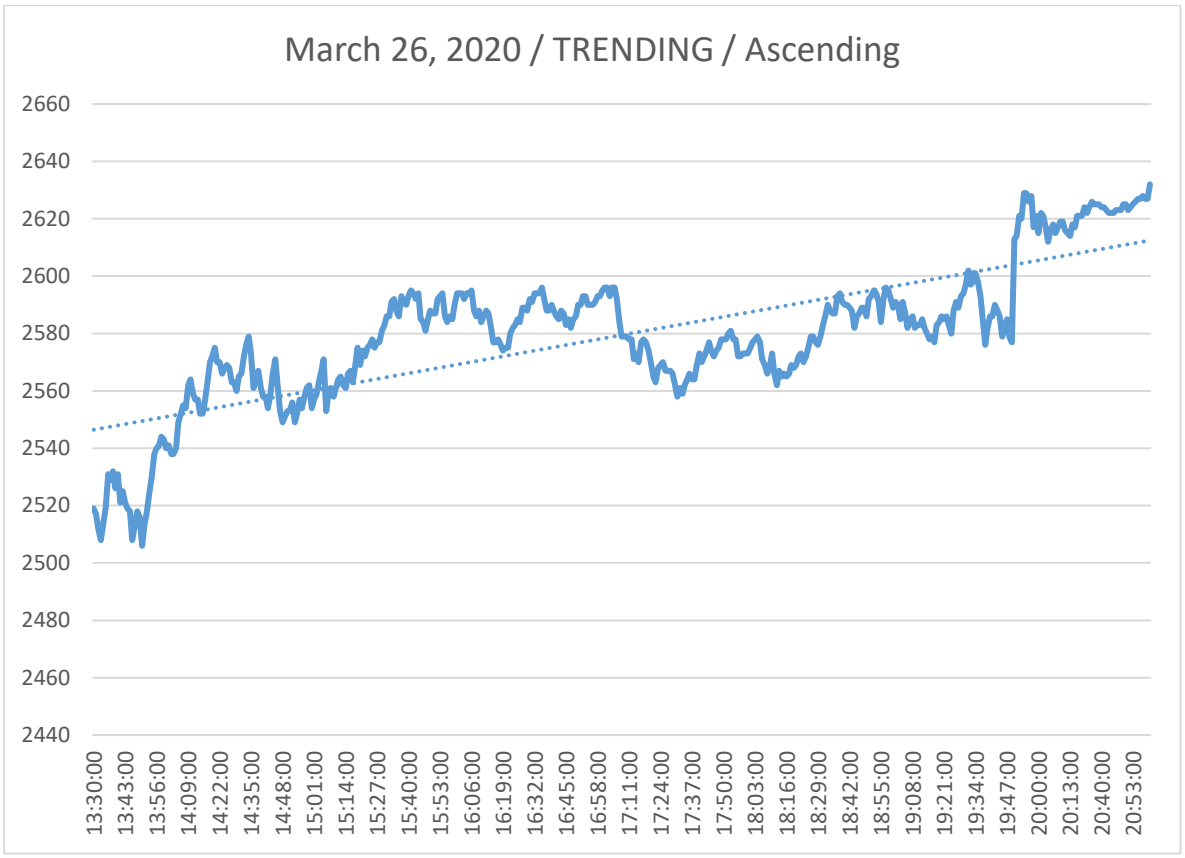

April 2, 2020 / TRENDING / RANGING / Ascending / Reversals
2540
2520
2500
2480
2460
2440
2420
2400
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:40:00
20:53:00

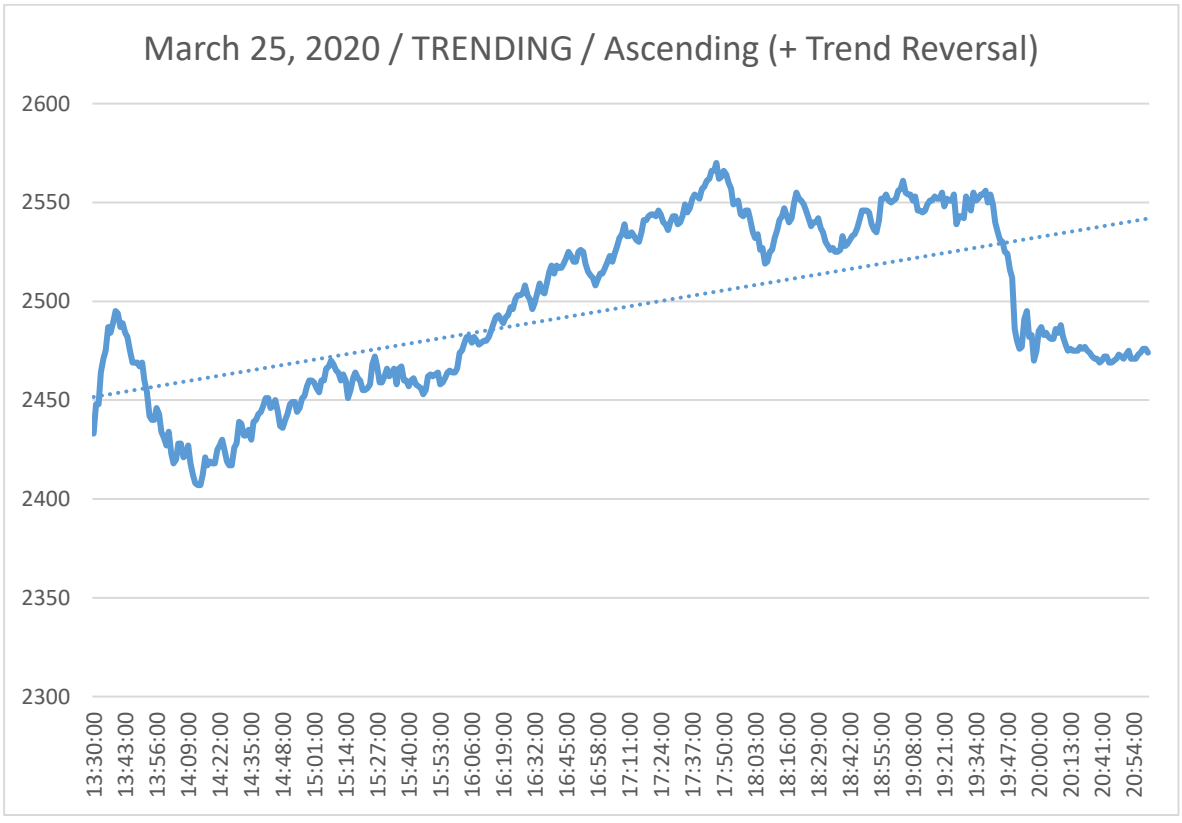

April 1, 2020 / TRENDING / Descending
2540
2520
2500
2480
2460
2440
2420
2400
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:40:00
20:53:00

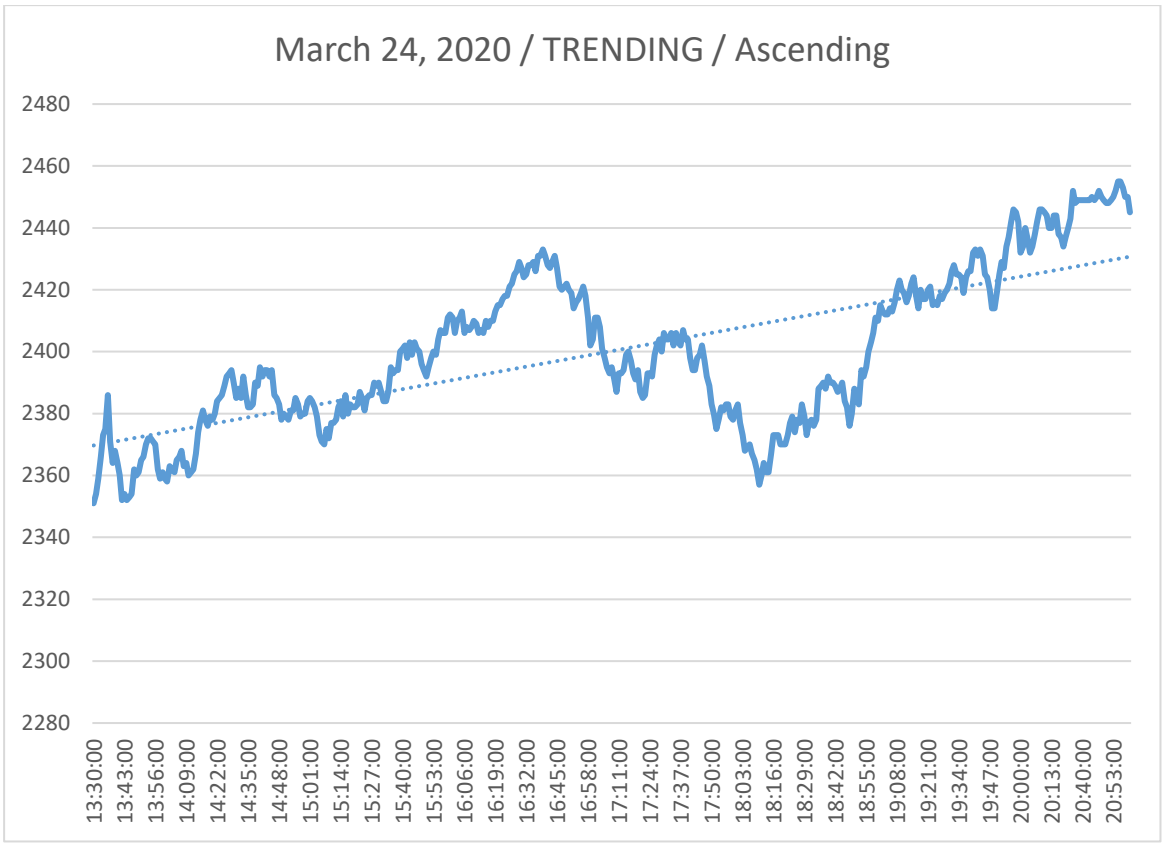

March 31, 2020 / TRENDING / Descending
2640
2620
2600
2580
2560
2540
2520
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:40:00
20:53:00

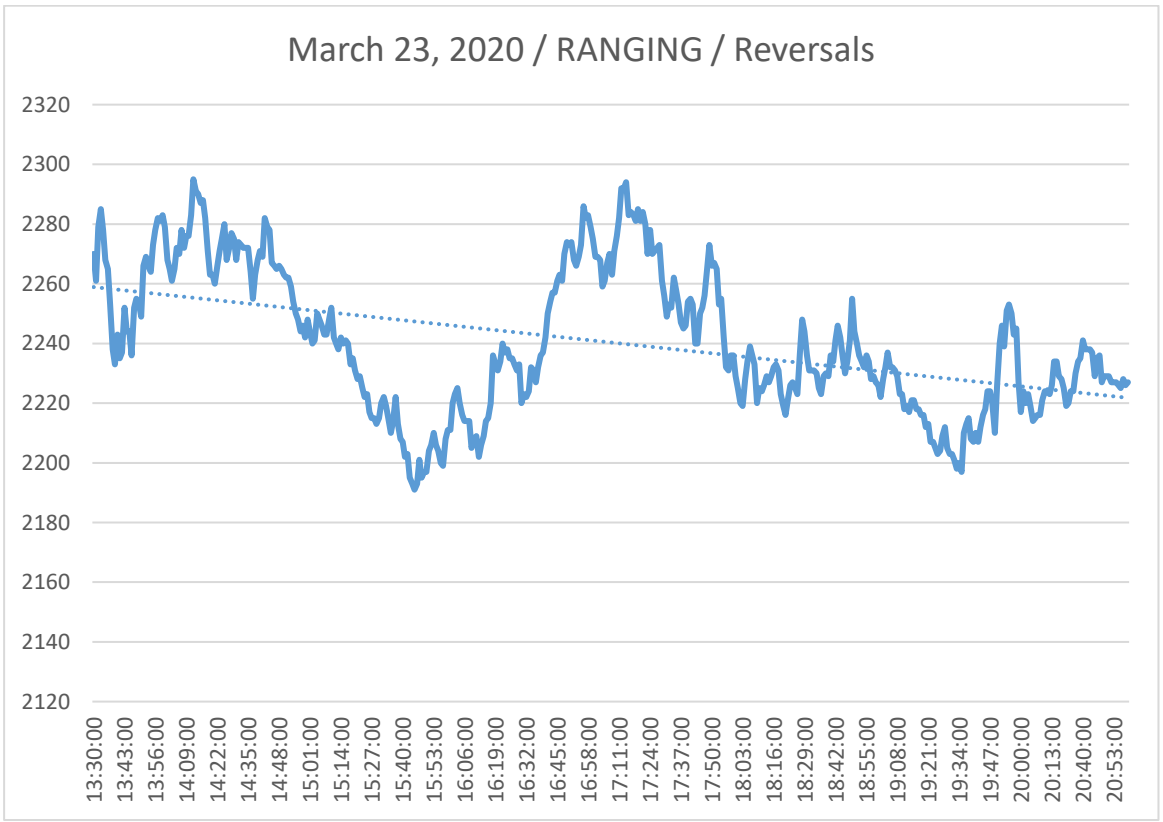

March 30, 2020 / TRENDING / Ascending
2640
2620
2600
2580
2560
2540
2520
2500
2480
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:40:00
20:53:00

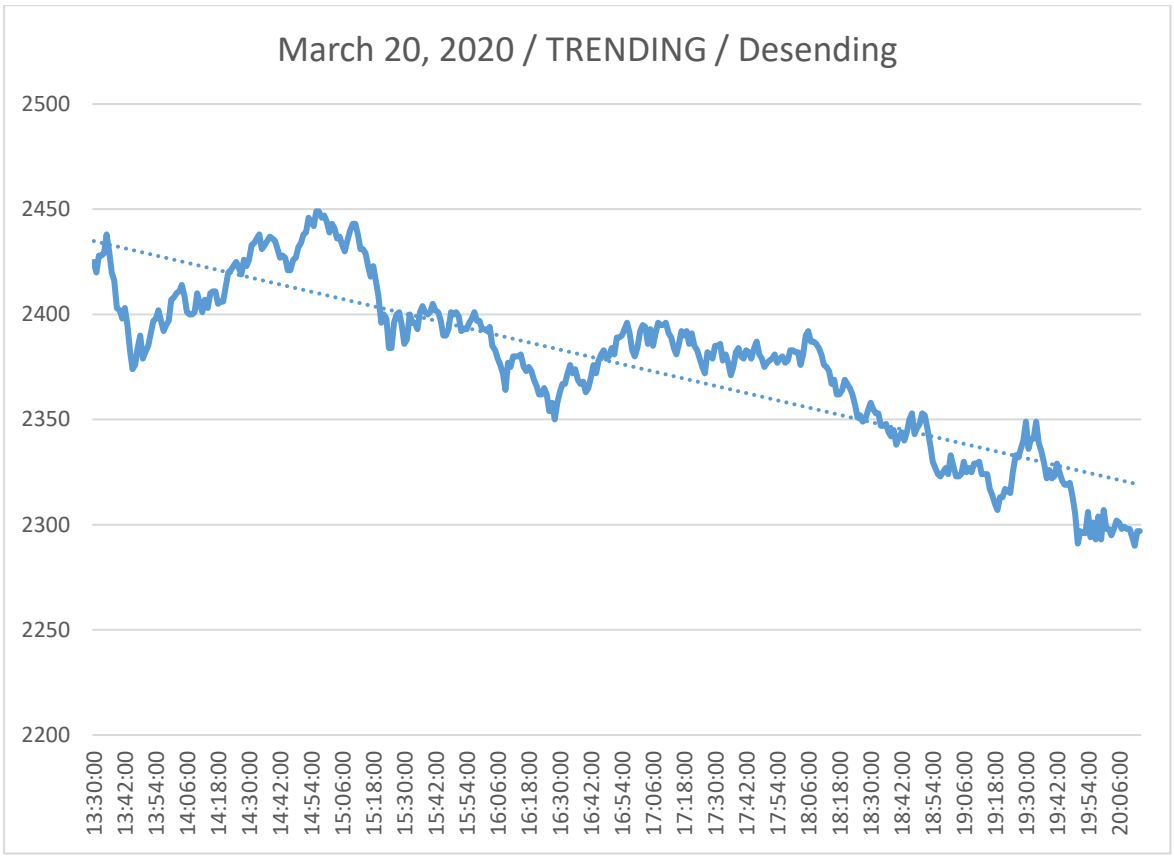

March 27, 2020 / TRENDING / Ascending (+Trend Reversal)
2620
2600
2580
2560
2540
2520
2500
2480
2460
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:30:00
14:42:00
14:54:00
15:06:00
15:18:00
15:30:00
15:42:00
15:54:00
16:06:00
16:18:00
16:30:00
16:42:00
16:54:00
17:06:00
17:18:00
17:30:00
17:42:00
17:54:00
18:06:00
18:18:00
18:30:00
18:42:00
18:54:00
19:06:00
19:18:00
19:30:00
19:42:00
19:54:00
20:06:00

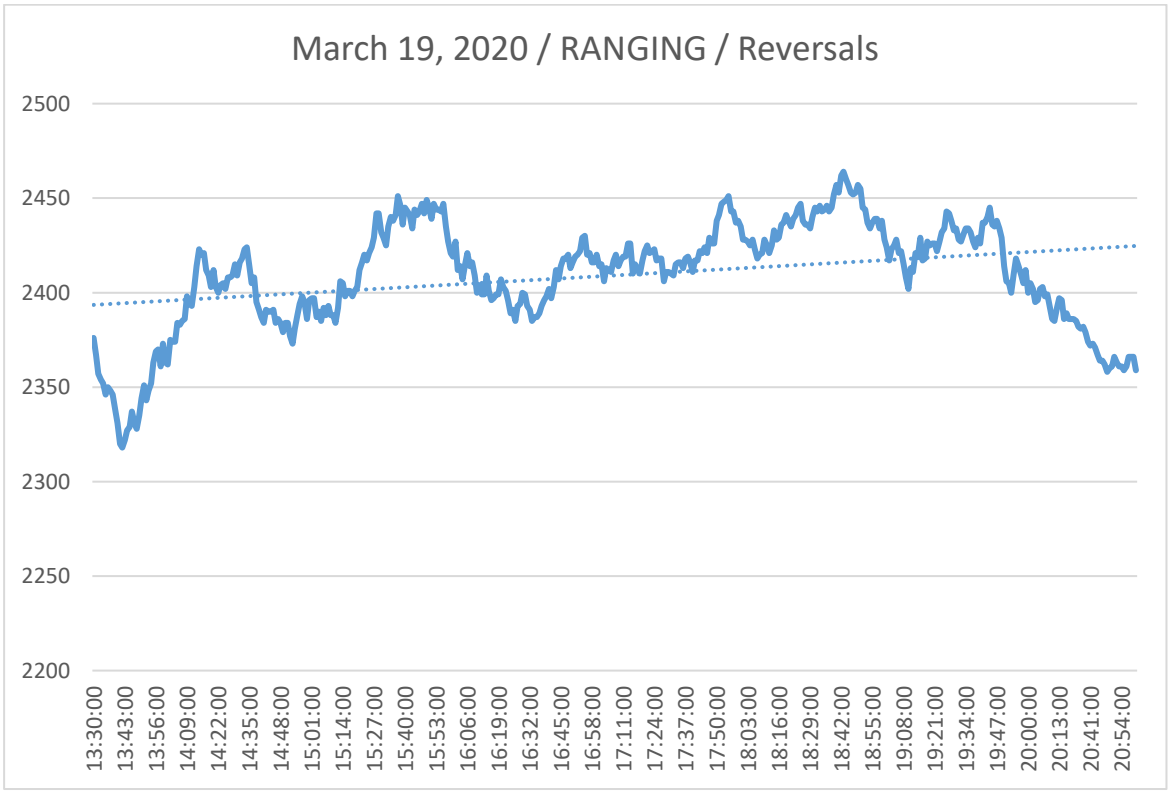

March 26, 2020 / TRENDING / Ascending
2660
2640
2620
2600
2580
2560
2540
2520
2500
2480
2460
2440
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:40:00
20:53:00

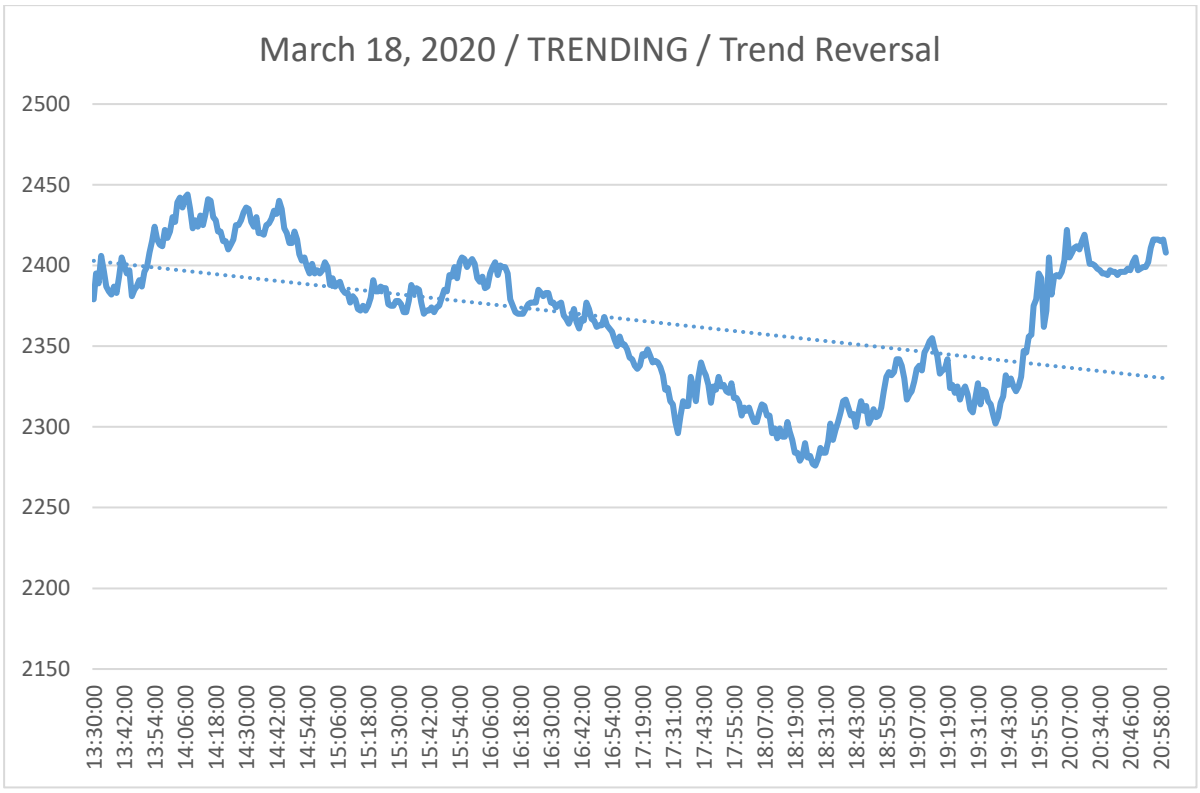

March 25, 2020 / TRENDING / Ascending (+ Trend Reversal)
2600
2550
2500
2450
2400
2350
2300
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

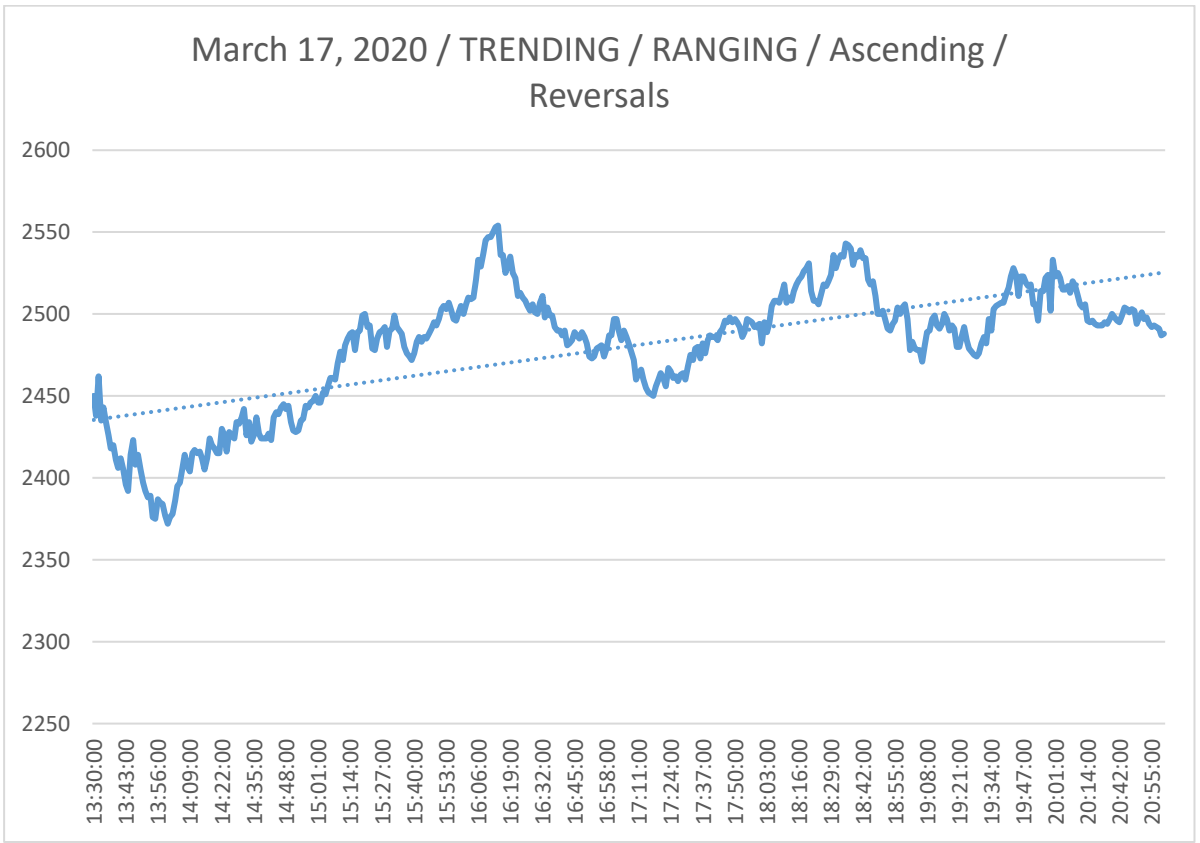

March 24, 2020 / TRENDING / Ascending
2480
2460
2440
2420
2400
2380
2360
2340
2320
2300
2280
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:40:00
20:53:00

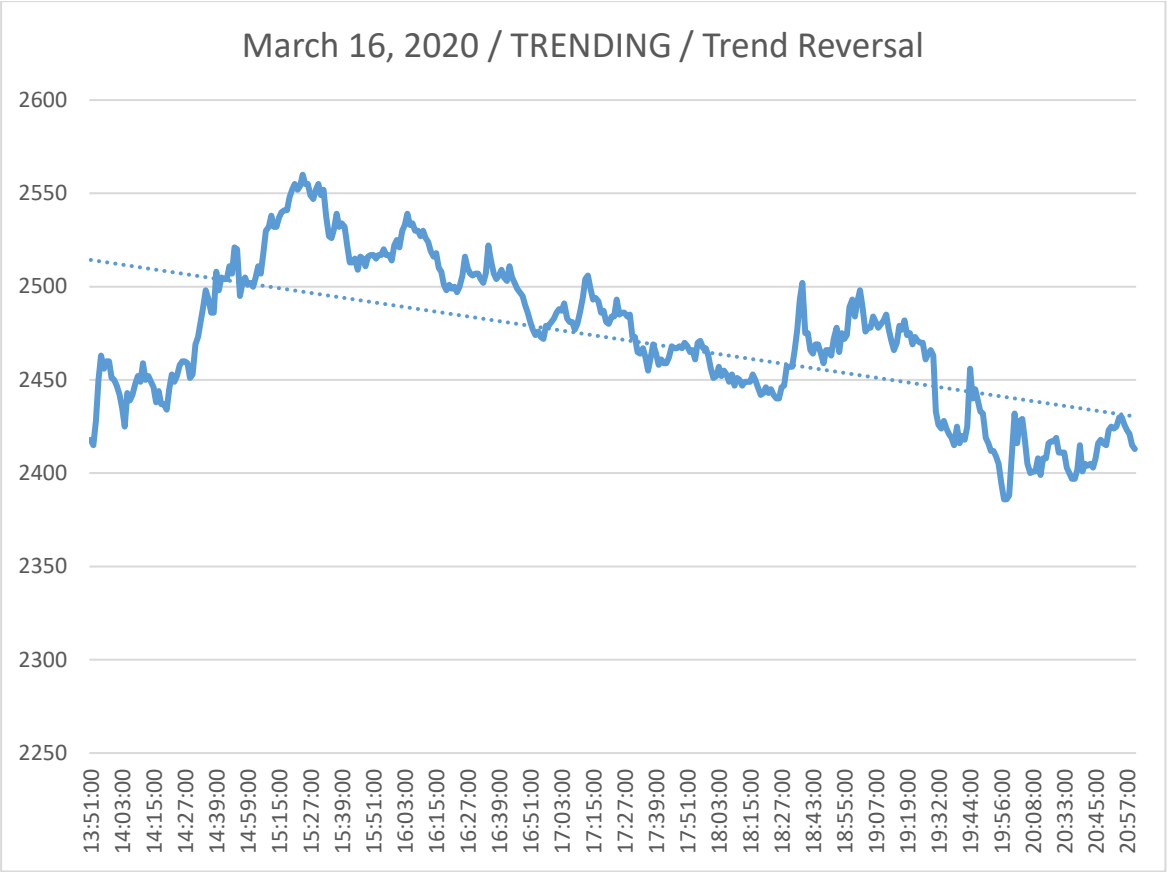

March 23, 2020 / RANGING / Reversals
2320
2300
2280
2260
2240
2220
2200
2180
2160
2140
2120
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:40:00
20:53:00

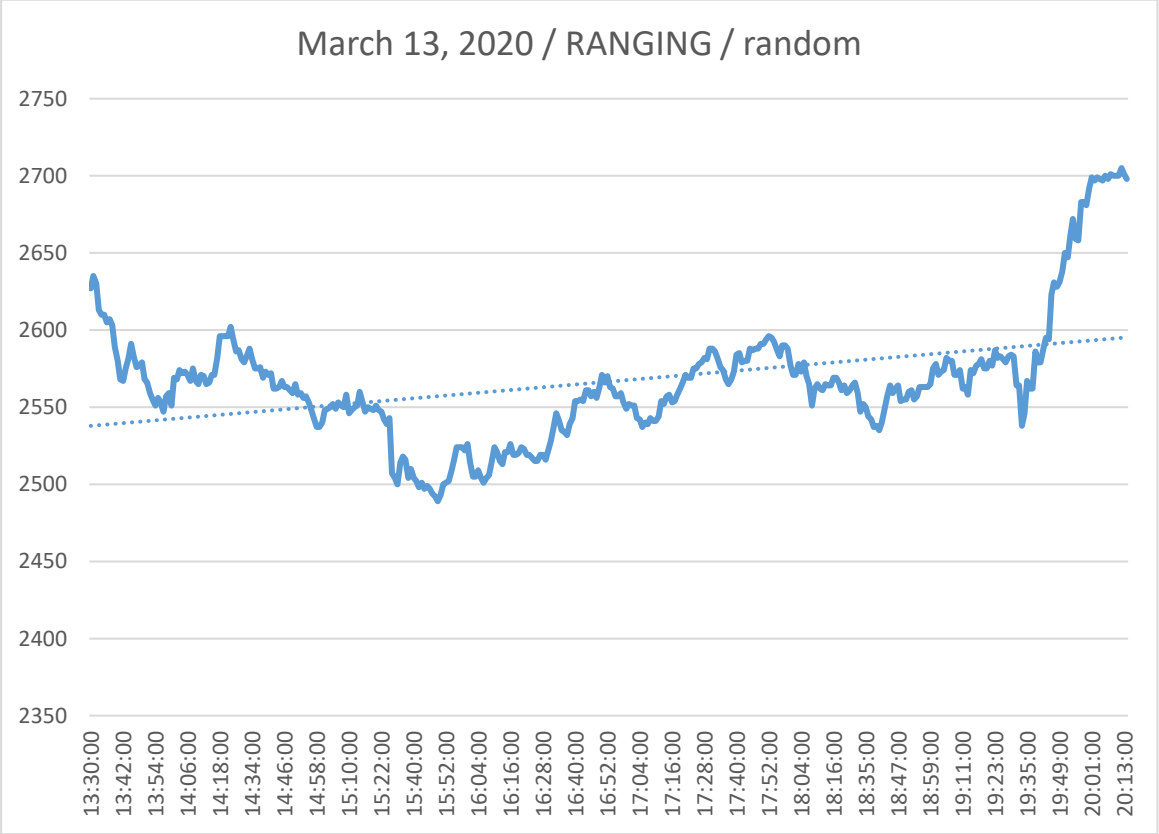

March 20, 2020 / TRENDING / Desending
2500
2450
2400
2350
2300
2250
2200
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:30:00
14:42:00
14:54:00
15:06:00
15:18:00
15:30:00
15:42:00
15:54:00
16:06:00
16:18:00
16:30:00
16:42:00
16:54:00
17:06:00
17:18:00
17:30:00
17:42:00
17:54:00
18:06:00
18:18:00
18:30:00
18:42:00
18:54:00
19:06:00
19:18:00
19:30:00
19:42:00
19:54:00
20:06:00

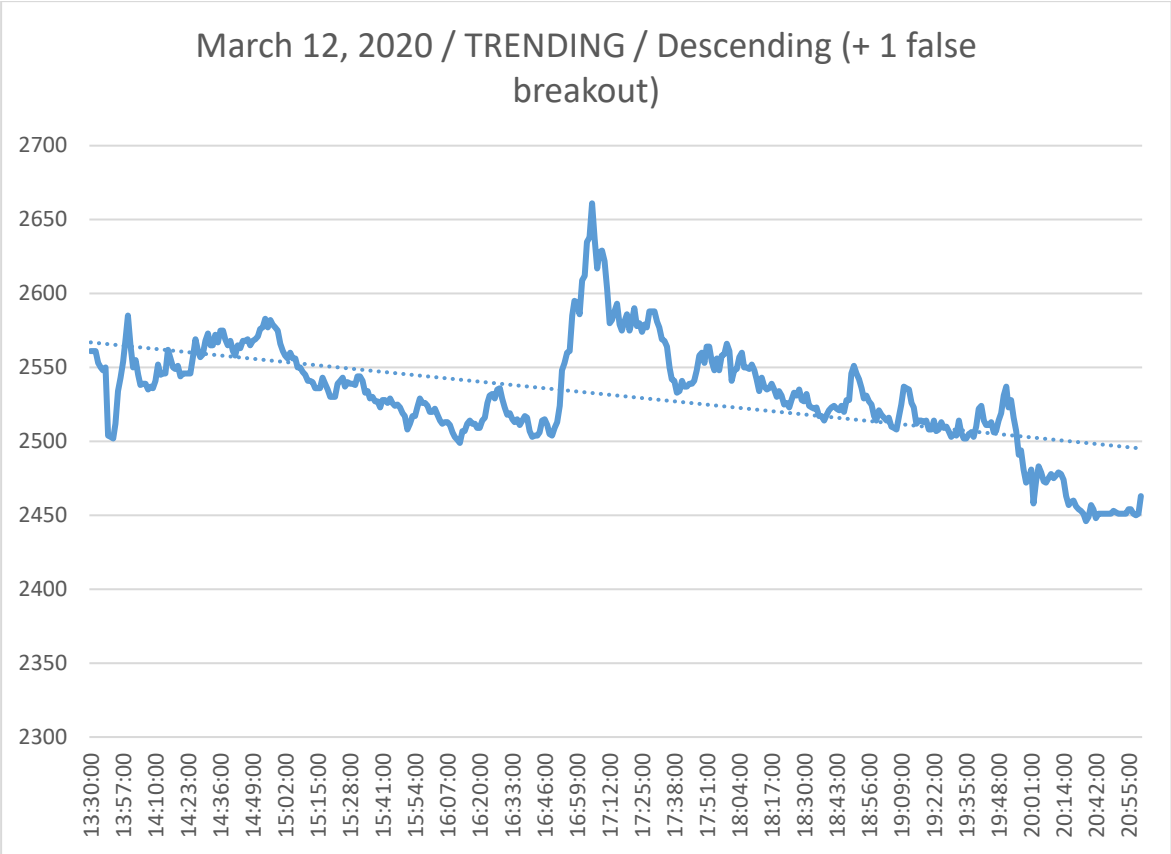

March 19, 2020 / RANGING / Reversals
2500
2450
2400
2350
2300
2250
2200
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

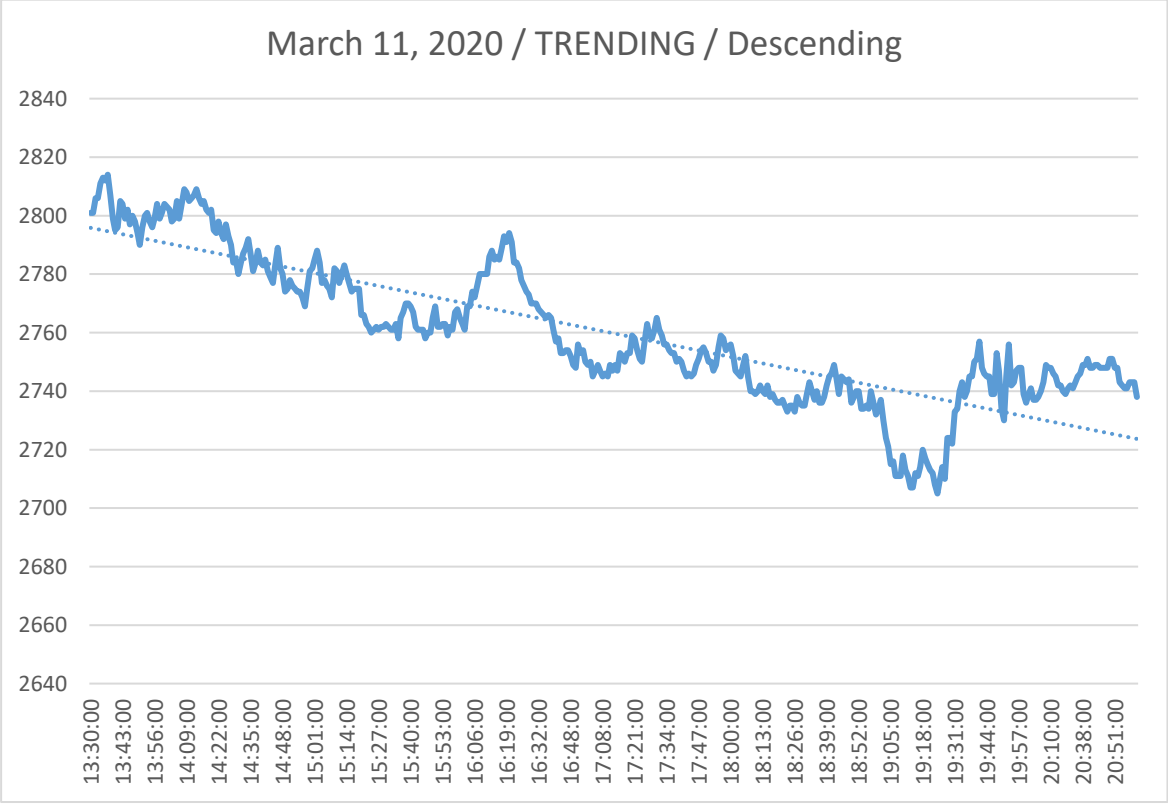

March 18, 2020 / TRENDING / Trend Reversal
2500
2450
2400
2350
2300
2250
2200
2150
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:30:00
14:42:00
14:54:00
15:06:00
15:18:00
15:30:00
15:42:00
15:54:00
16:06:00
16:18:00
16:30:00
16:42:00
16:54:00
17:19:00
17:31:00
17:43:00
17:55:00
18:07:00
18:19:00
18:31:00
18:43:00
18:55:00
19:07:00
19:19:00
19:31:00
19:43:00
19:55:00
20:07:00
20:34:00
20:46:00
20:58:00

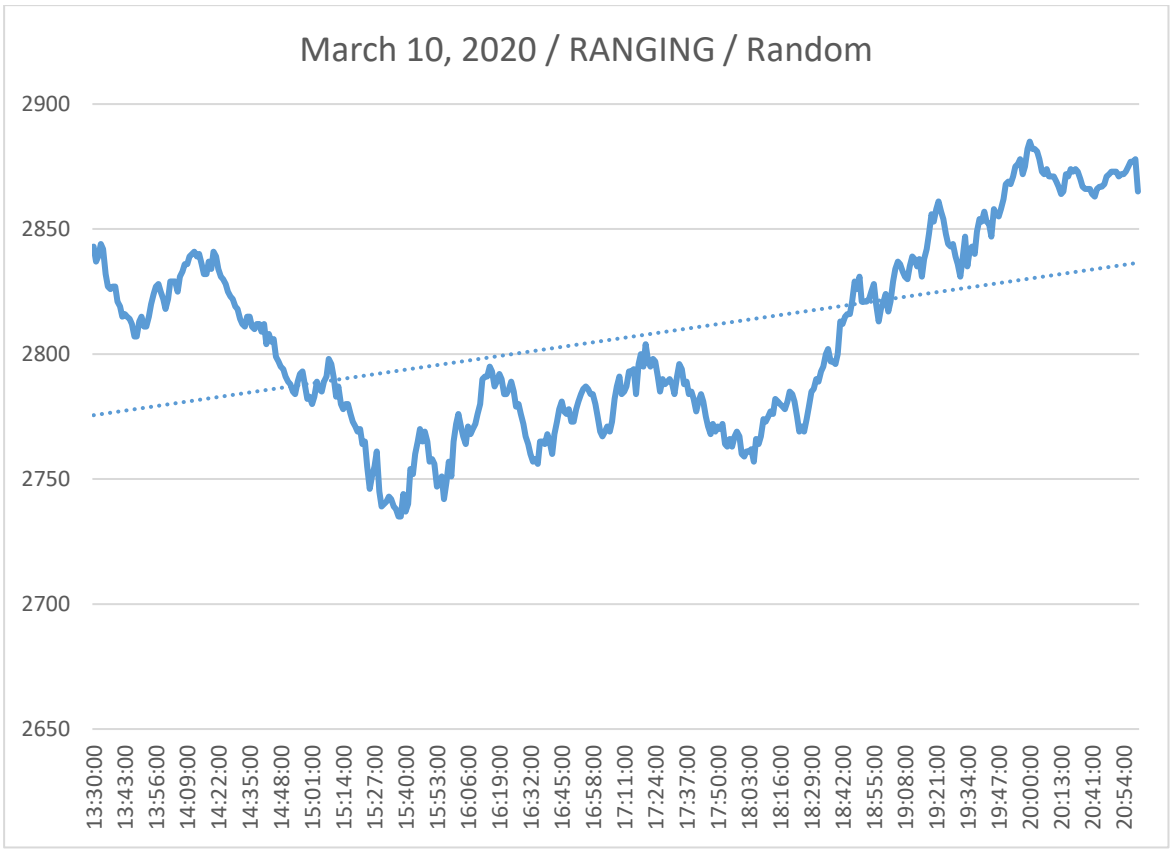

March 17, 2020 / TRENDING / RANGING / Ascending / Reversals
2600
2550
2500
2450
2400
2350
2300
2250
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:01:00
20:14:00
20:42:00
20:55:00

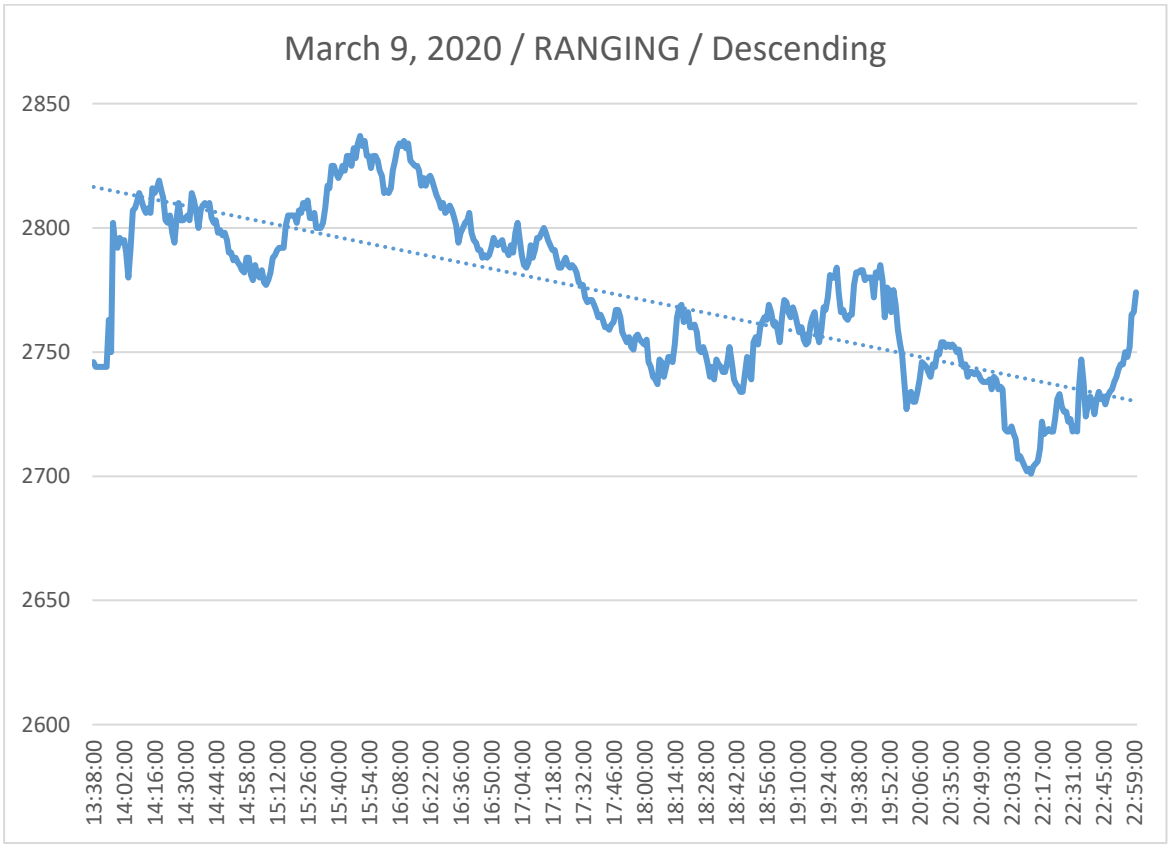

March 16, 2020 / TRENDING / Trend Reversal
2600
2550
2500
2450
2400
2350
2300
2250
13:51:00
14:03:00
14:15:00
14:27:00
14:39:00
14:59:00
15:15:00
15:27:00
15:39:00
15:51:00
16:03:00
16:15:00
16:27:00
16:39:00
16:51:00
17:03:00
17:15:00
17:27:00
17:39:00
17:51:00
18:03:00
18:15:00
18:27:00
18:43:00
18:55:00
19:07:00
19:19:00
19:32:00
19:44:00
19:56:00
20:08:00
20:33:00
20:45:00
20:57:00

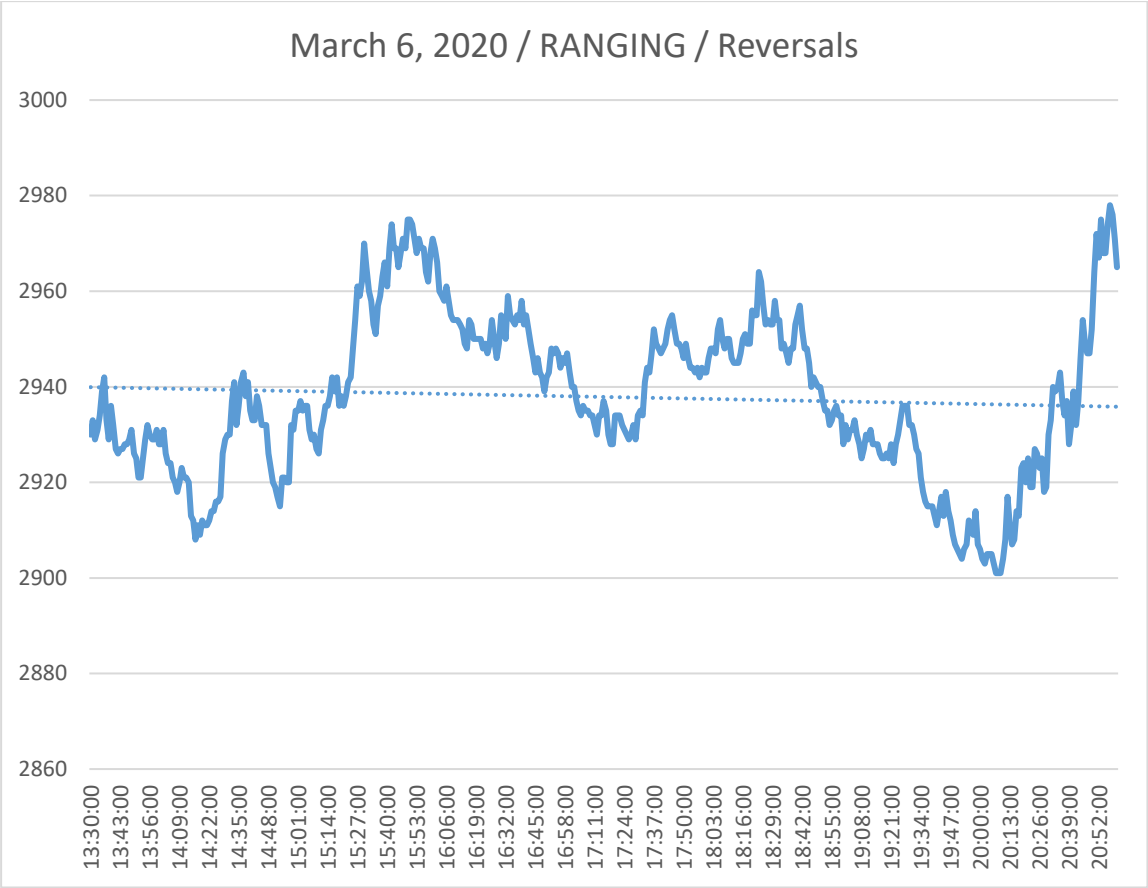

March 13, 2020 / RANGING / random
2750
2700
2650
2600
2550
2500
2450
2400
2350
13:30:00
13:42:00
13:54:00
14:06:00
14:18:00
14:34:00
14:46:00
14:58:00
15:10:00
15:22:00
15:40:00
15:52:00
16:04:00
16:16:00
16:28:00
16:40:00
16:52:00
17:04:00
17:16:00
17:28:00
17:40:00
17:52:00
18:04:00
18:16:00
18:35:00
18:47:00
18:59:00
19:11:00
19:23:00
19:35:00
19:49:00
20:01:00
20:13:00

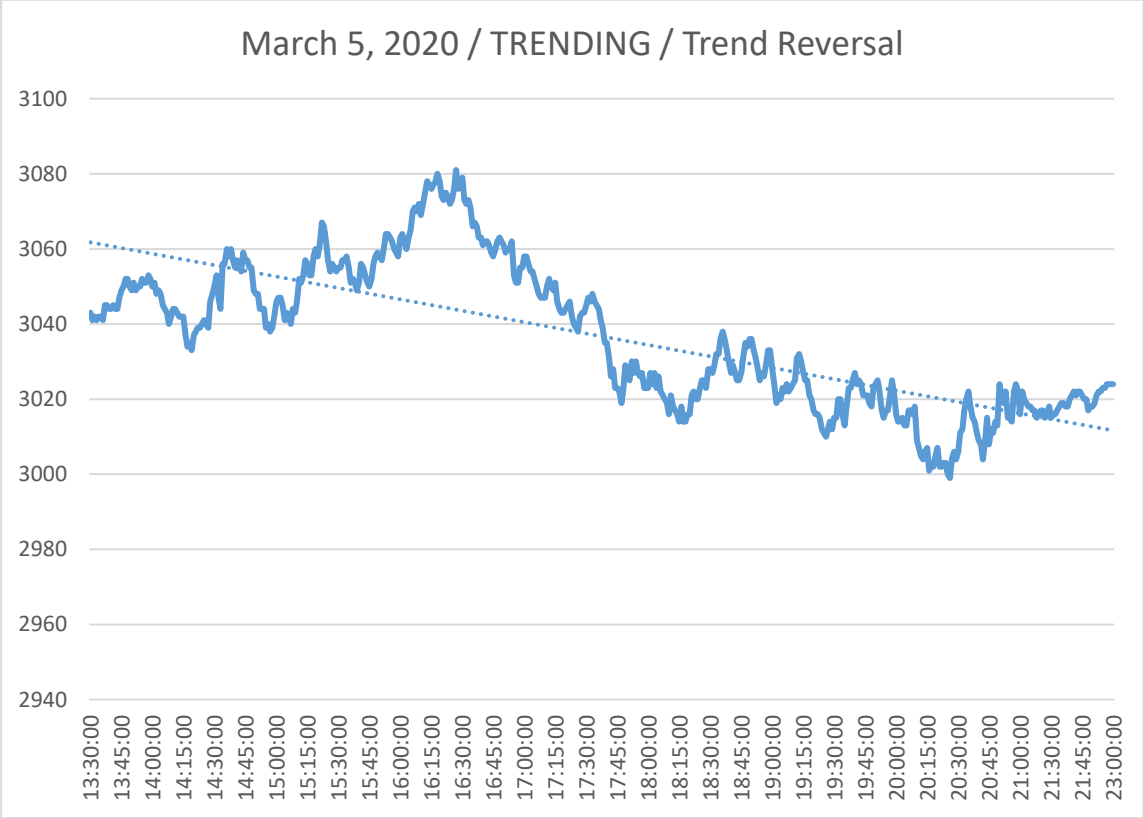

March 12, 2020 / TRENDING / Descending (+ 1 false breakout)
2700
2650
2600
2550
2500
2450
2400
2350
2300
13:30:00
13:57:00
14:10:00
14:23:00
14:36:00
14:49:00
15:02:00
15:15:00
15:28:00
15:41:00
15:54:00
16:07:00
16:20:00
16:33:00
16:46:00
16:59:00
17:12:00
17:25:00
17:38:00
17:51:00
18:04:00
18:17:00
18:30:00
18:43:00
18:56:00
19:09:00
19:22:00
19:35:00
19:48:00
20:01:00
20:14:00
20:42:00
20:55:00

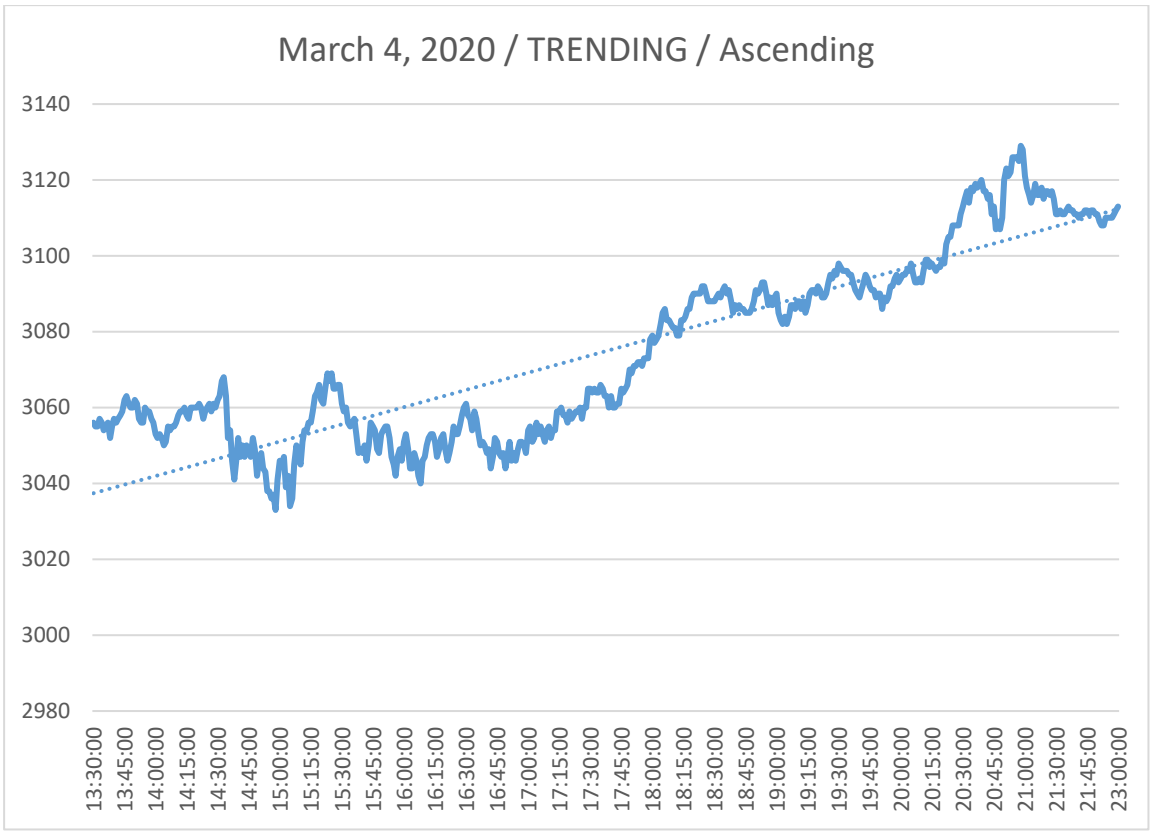

March 11, 2020 / TRENDING / Descending
2840
2820
2800
2780
2760
2740
2720
2700
2680
2660
2640
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:48:00
17:08:00
17:21:00
17:34:00
17:47:00
18:00:00
18:13:00
18:26:00
18:39:00
18:52:00
19:05:00
19:18:00
19:31:00
19:44:00
19:57:00
20:10:00
20:38:00
20:51:00

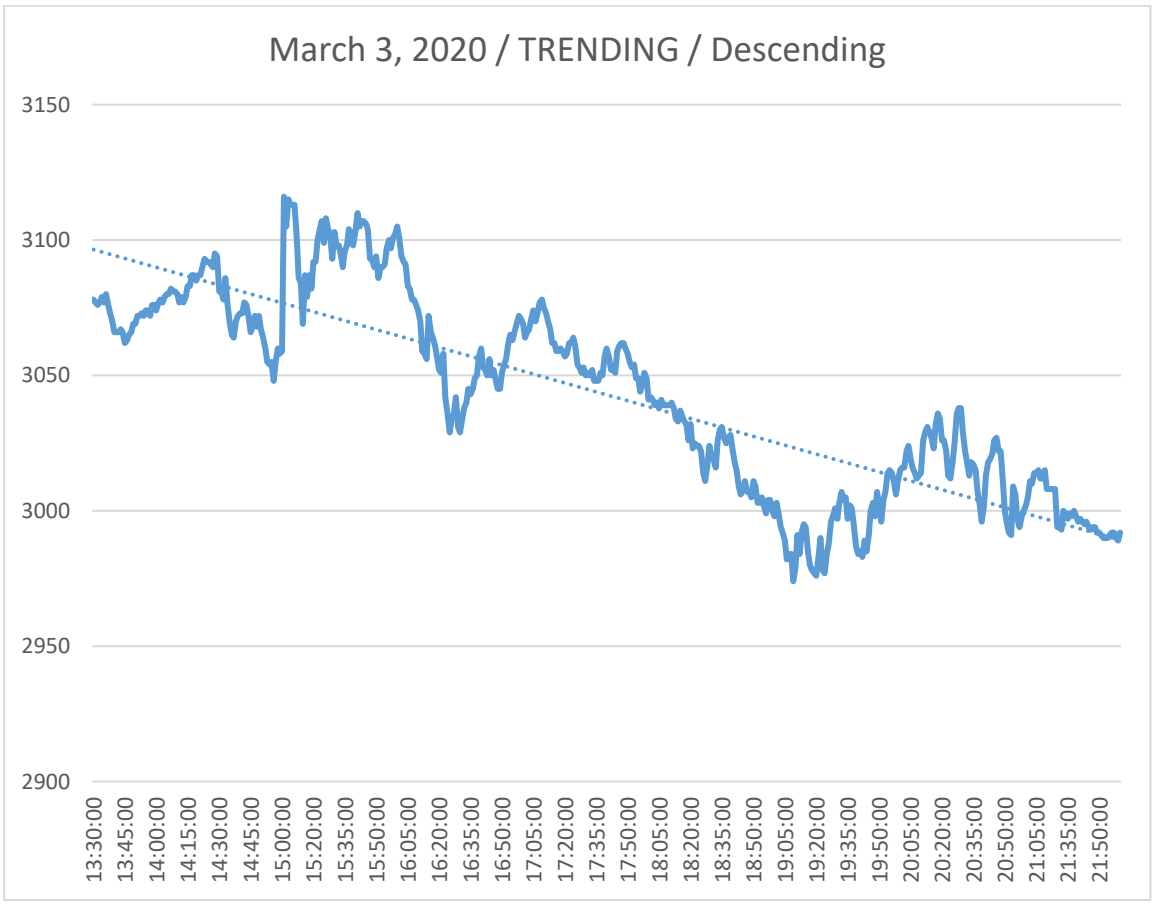

March 10, 2020 / RANGING / Random
2900
2850
2800
2750
2700
2650
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:41:00
20:54:00

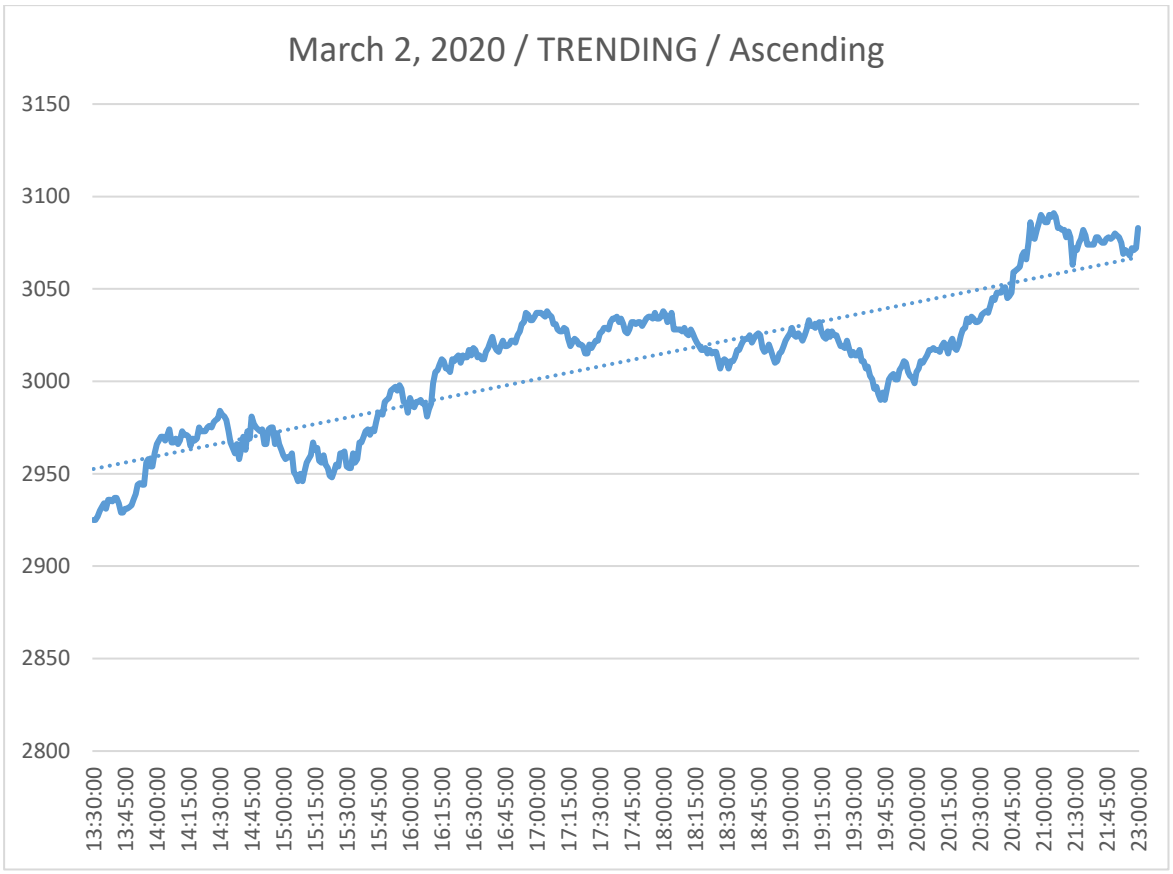

March 9, 2020 / RANGING / Descending
2850
2800
2750
2700
2650
2600
13:38:00
14:02:00
14:16:00
14:30:00
14:44:00
14:58:00
15:12:00
15:26:00
15:40:00
15:54:00
16:08:00
16:22:00
16:36:00
16:50:00
17:04:00
17:18:00
17:32:00
17:46:00
18:00:00
18:14:00
18:28:00
18:42:00
18:56:00
19:10:00
19:24:00
19:38:00
19:52:00
20:06:00
20:35:00
20:49:00
22:03:00
22:17:00
22:31:00
22:45:00
22:59:00

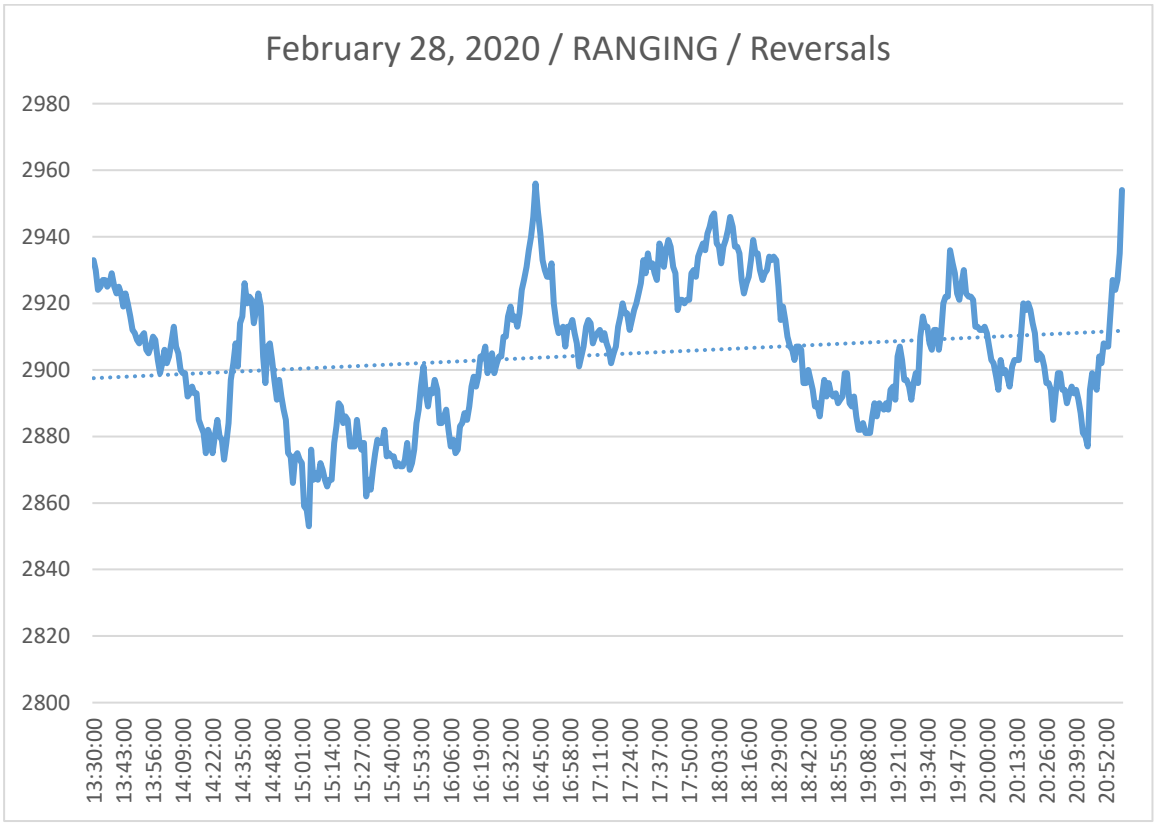

March 6, 2020 / RANGING / Reversals
3000
2980
2960
2940
2920
2900
2880
2860
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:26:00
20:39:00
20:52:00

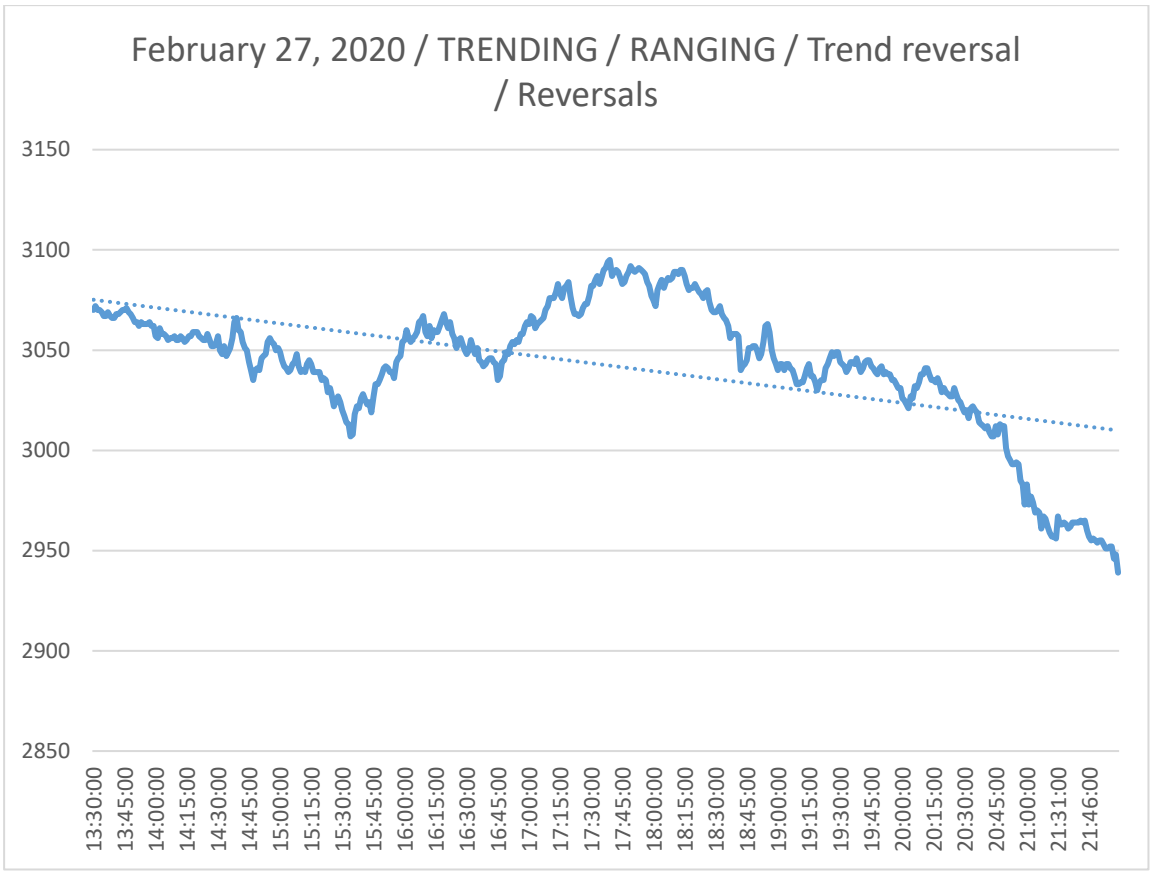

March 5, 2020 / TRENDING / Trend Reversal
3100
3080
3060
3040
3020
3000
2980
2960
2940
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00

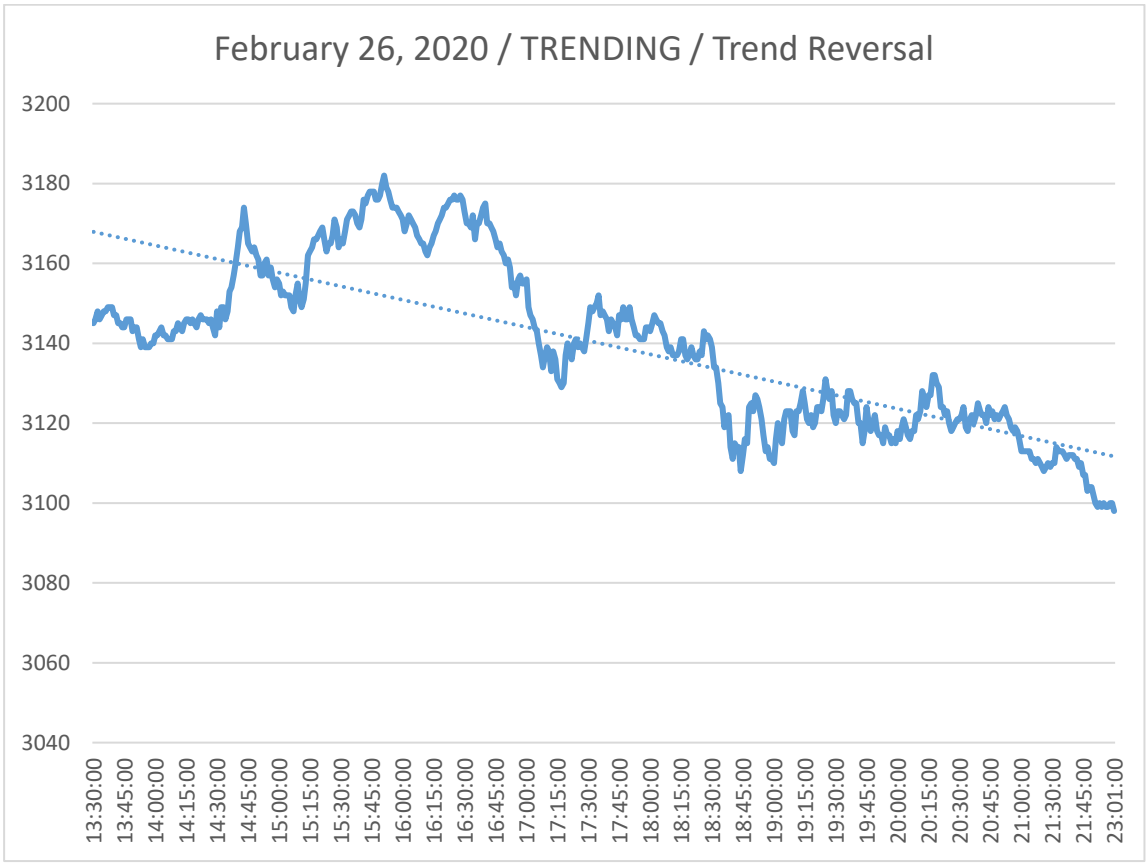

March 4, 2020 / TRENDING / Ascending
3140
3120
3100
3080
3060
3040
3020
3000
2980
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00

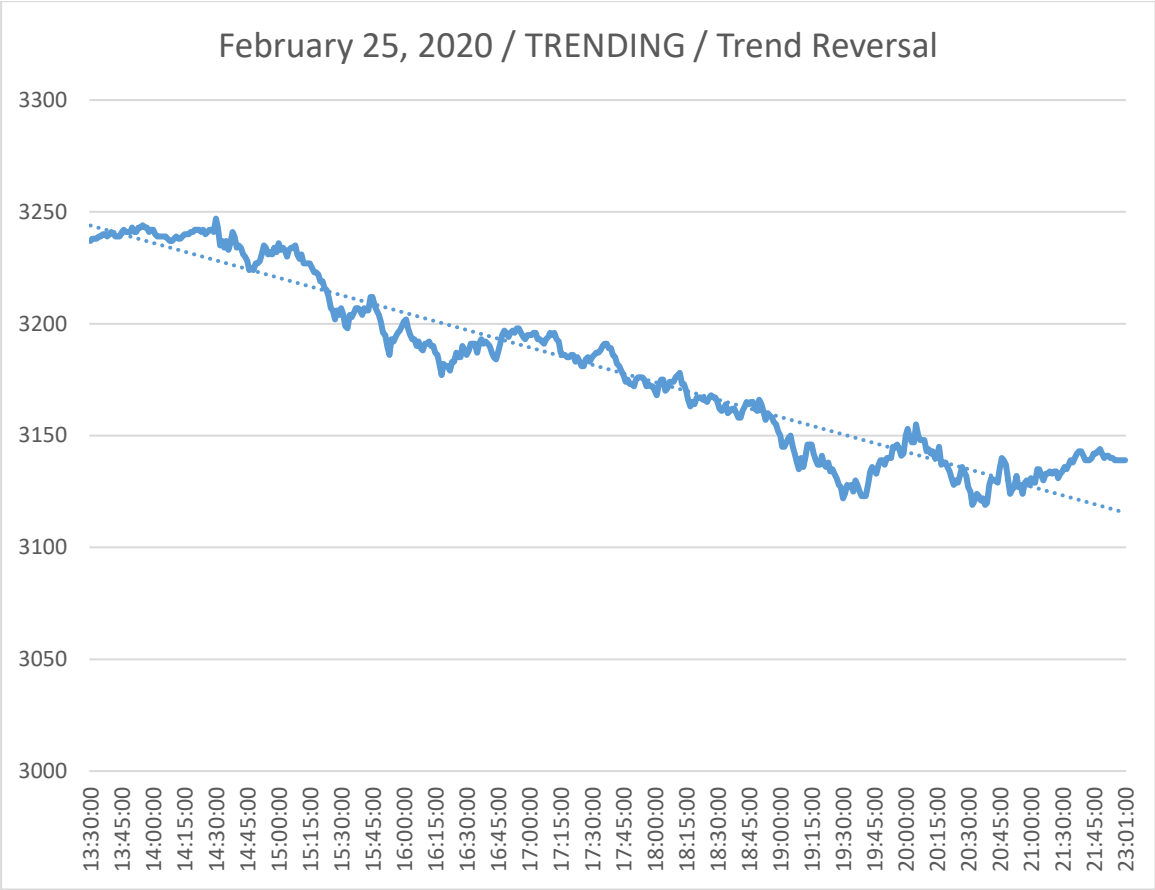

March 3, 2020 / TRENDING / Descending
3150
3100
3050
3000
2950
2900
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:20:00
15:35:00
15:50:00
16:05:00
16:20:00
16:35:00
16:50:00
17:05:00
17:20:00
17:35:00
17:50:00
18:05:00
18:20:00
18:35:00
18:50:00
19:05:00
19:20:00
19:35:00
19:50:00
20:05:00
20:20:00
20:35:00
20:50:00
21:05:00
21:35:00
21:50:00

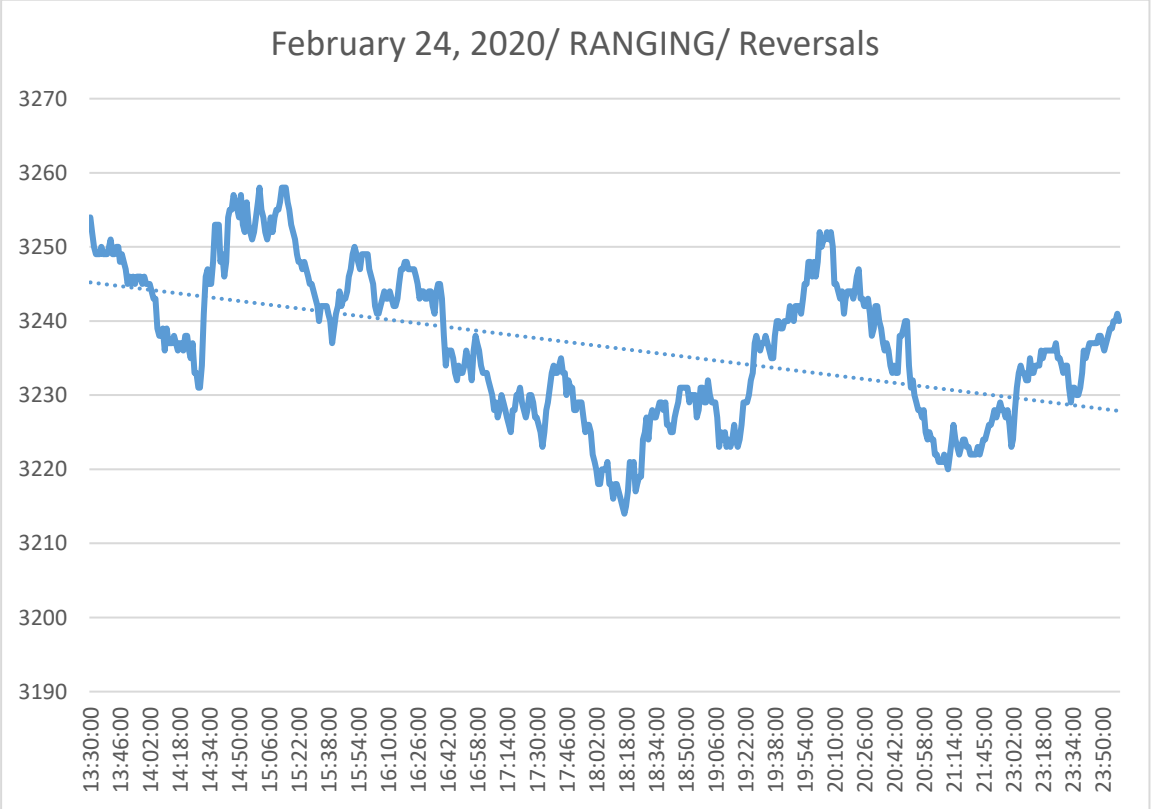

March 2, 2020 / TRENDING / Ascending
3150
3100
3050
3000
2950
2900
2850
2800
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00

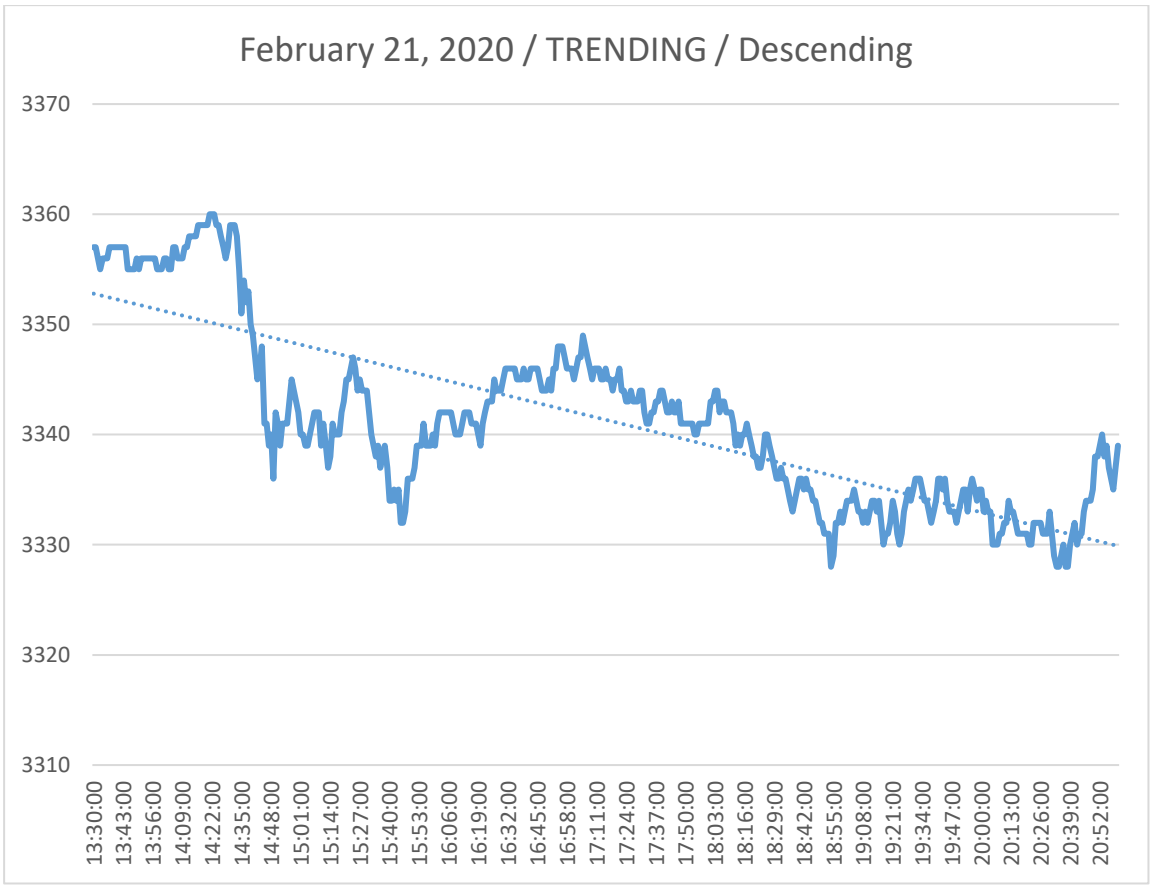

February 28, 2020 / RANGING / Reversals
2980
2960
2940
2920
2900
2880
2860
2840
2820
2800
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:26:00
20:39:00
20:52:00

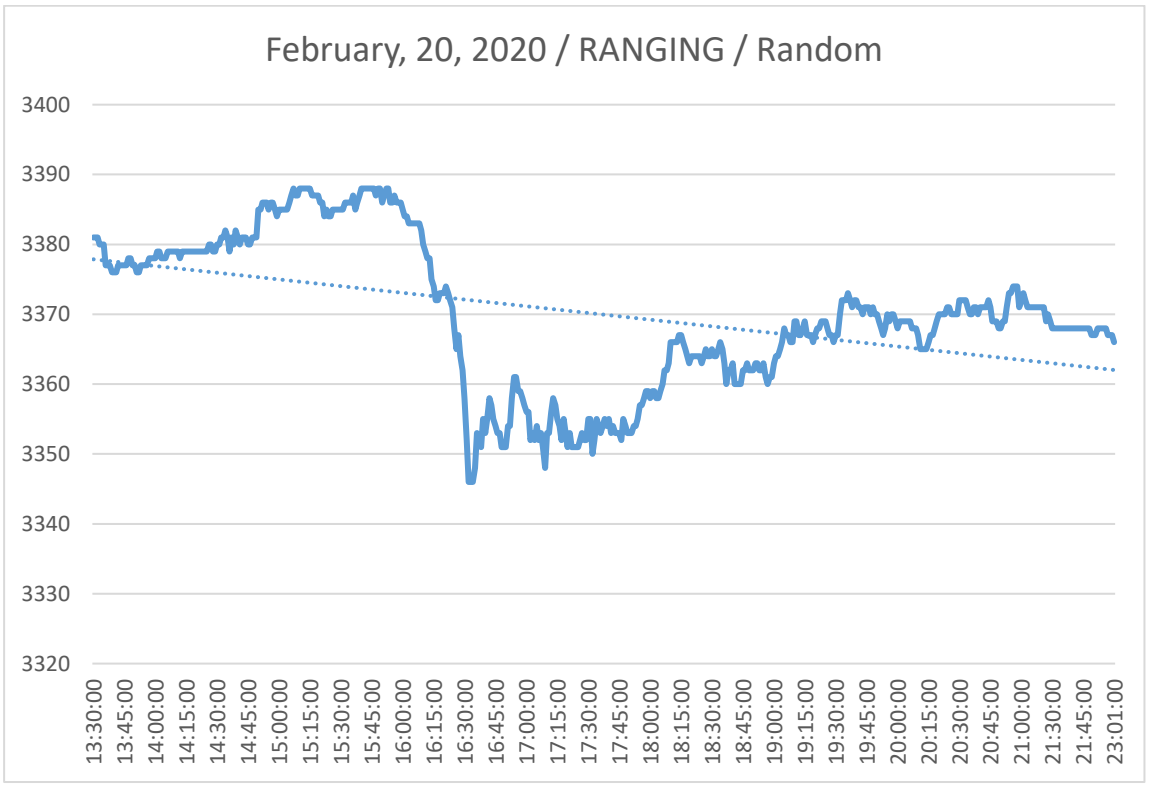

February 27, 2020 / TRENDING / RANGING / Trend reversal / Reversals
3150
3100
3050
3000
2950
2900
2850
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:31:00
21:46:00

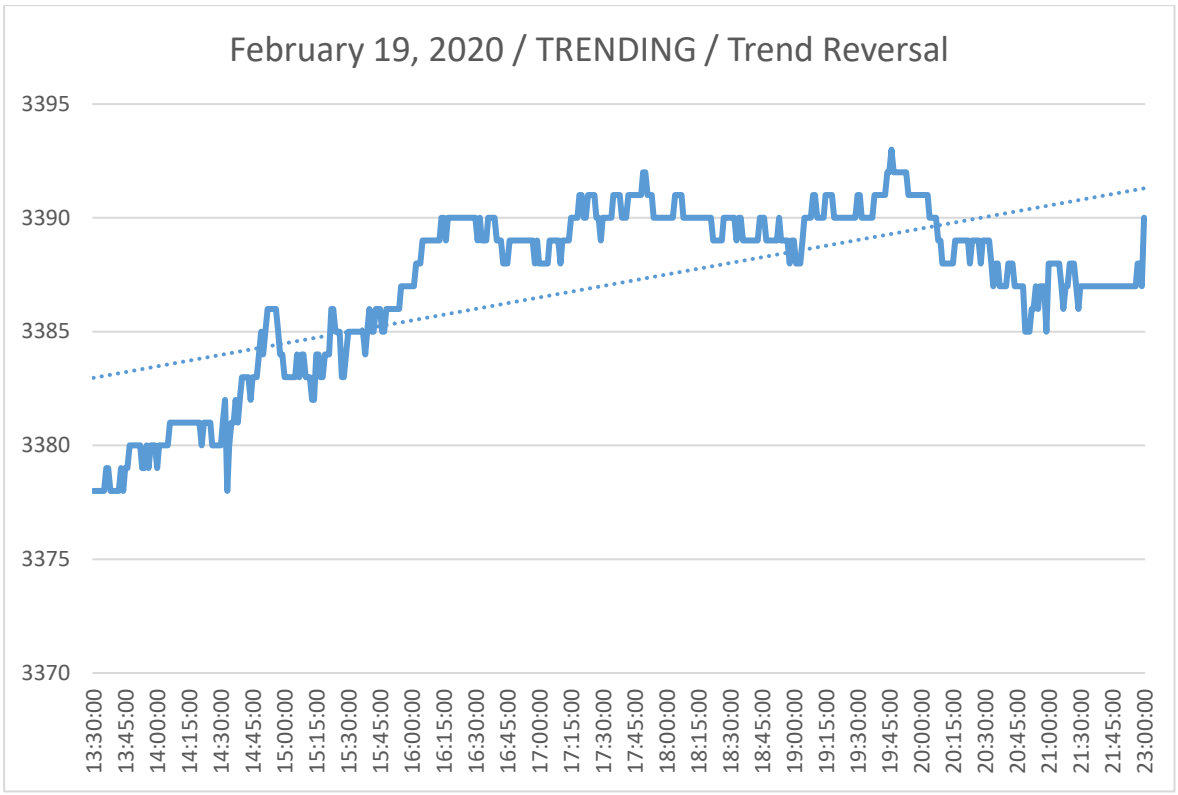

February 26, 2020 / TRENDING / Trend Reversal
3200
3180
3160
3140
3120
3100
3080
3060
3040
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:01:00

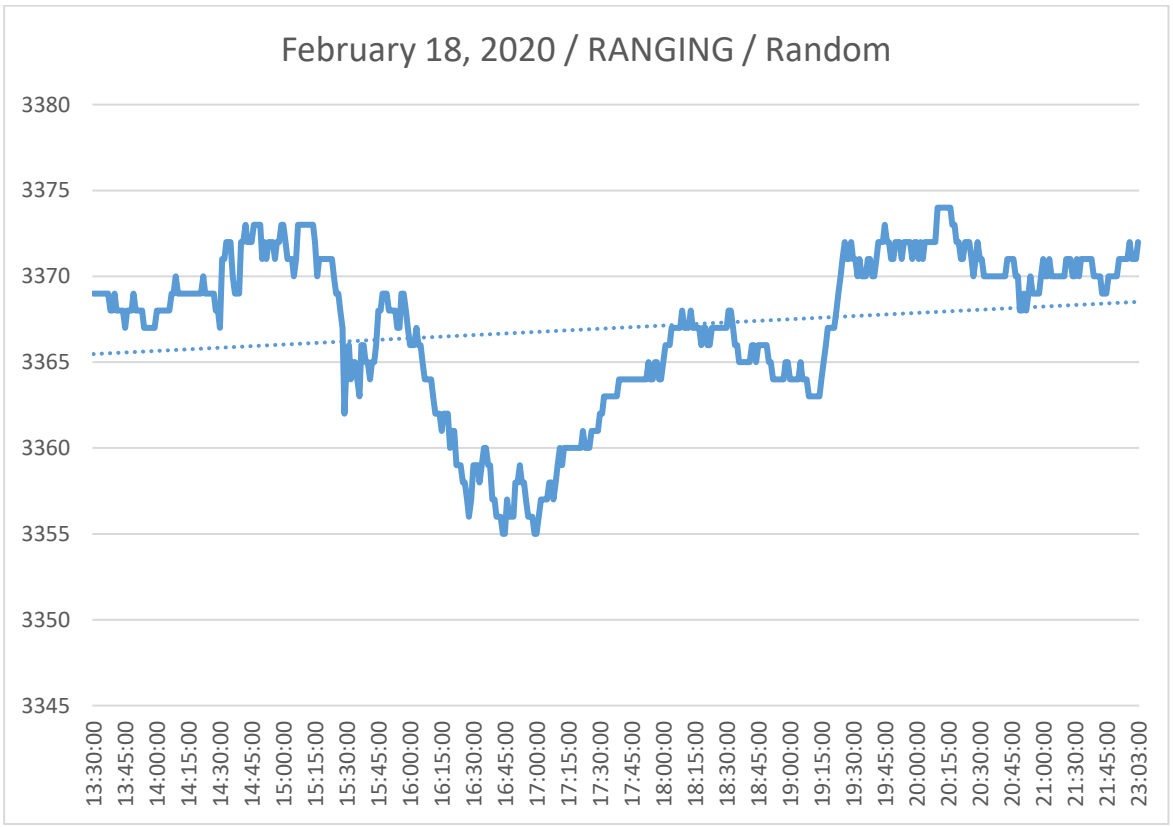

February 25, 2020 / TRENDING / Trend Reversal
3300
3250
3200
3150
3100
3050
3000
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:01:00

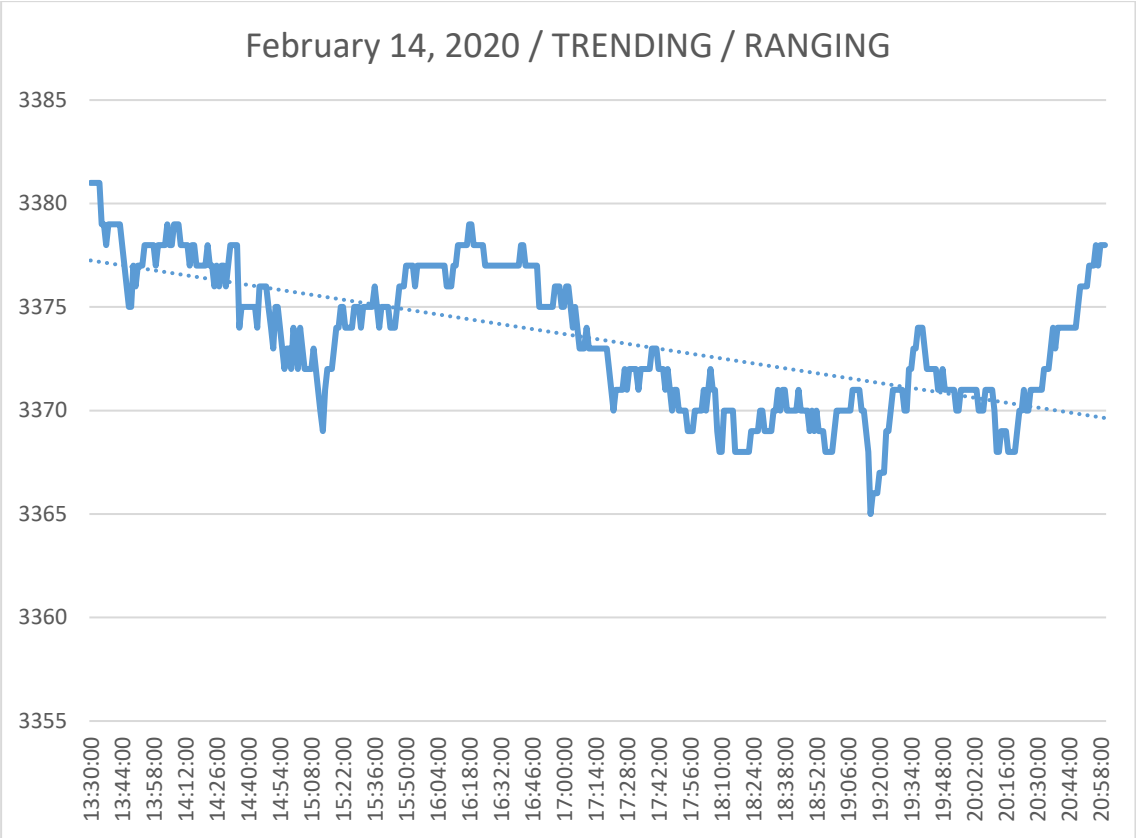

February 24, 2020/ RANGING/ Reversals
3270
3260
3250
3240
3230
3220
3210
3200
3190
13:30:00
13:46:00
14:02:00
14:18:00
14:34:00
14:50:00
15:06:00
15:22:00
15:38:00
15:54:00
16:10:00
16:26:00
16:42:00
16:58:00
17:14:00
17:30:00
17:46:00
18:02:00
18:18:00
18:34:00
18:50:00
19:06:00
19:22:00
19:38:00
19:54:00
20:10:00
20:26:00
20:42:00
20:58:00
21:14:00
21:45:00
23:02:00
23:18:00
23:34:00
23:50:00

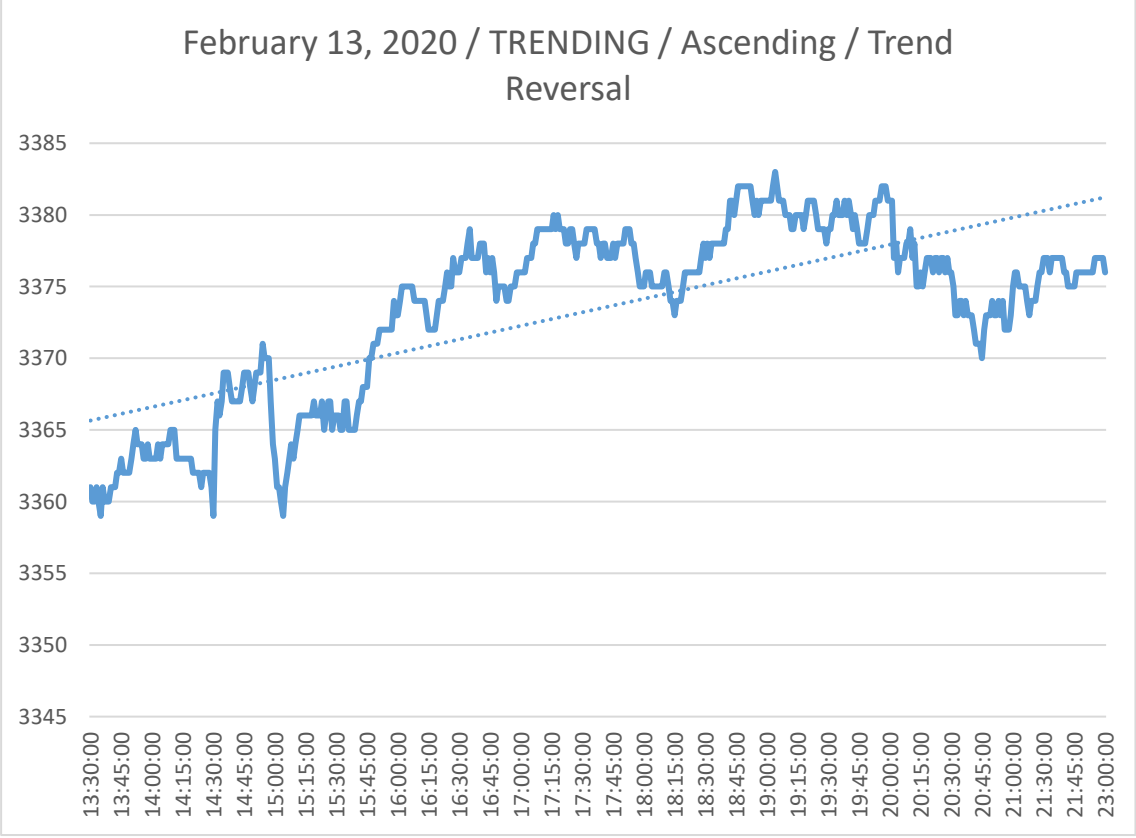

February 21, 2020 / TRENDING / Descending
3370
3360
3350
3340
3330
3320
3310
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:26:00
20:39:00
20:52:00

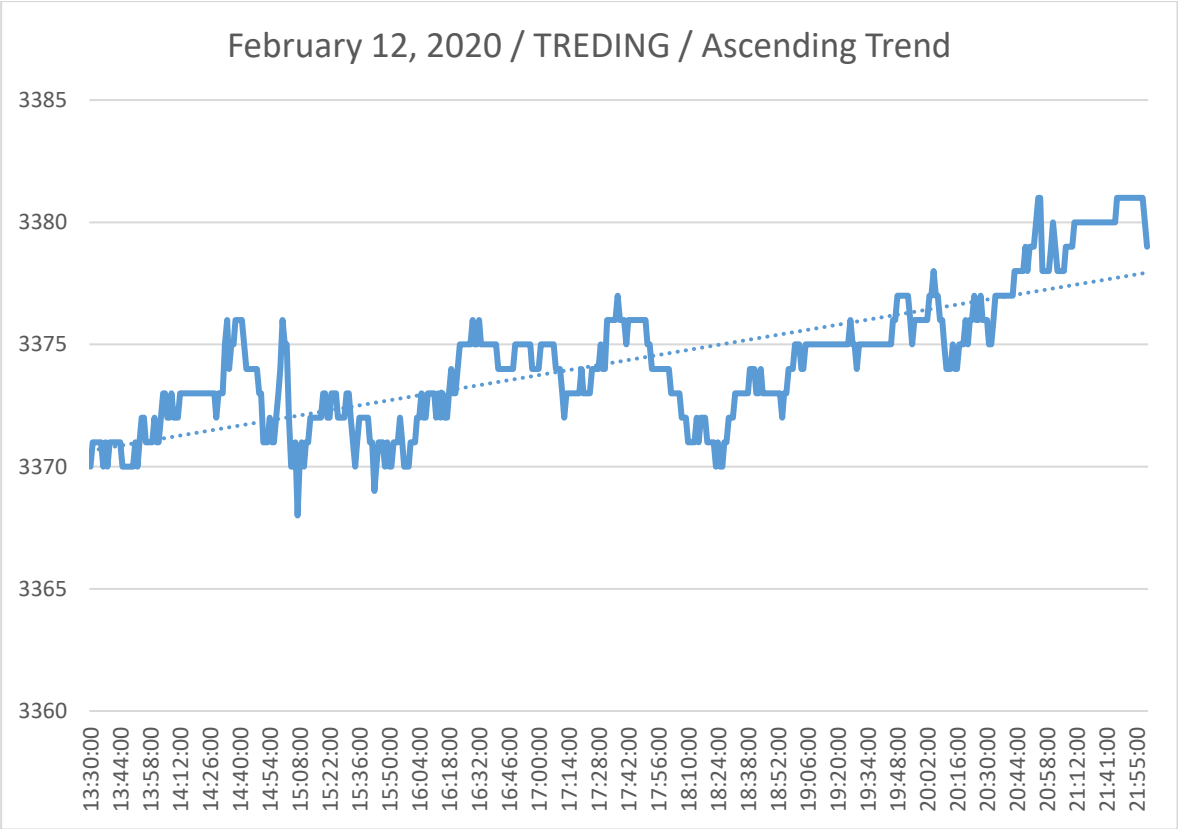

February, 20, 2020 / RANGING / Random
3400
3390
3380
3370
3360
3350
3340
3330
3320
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:01:00

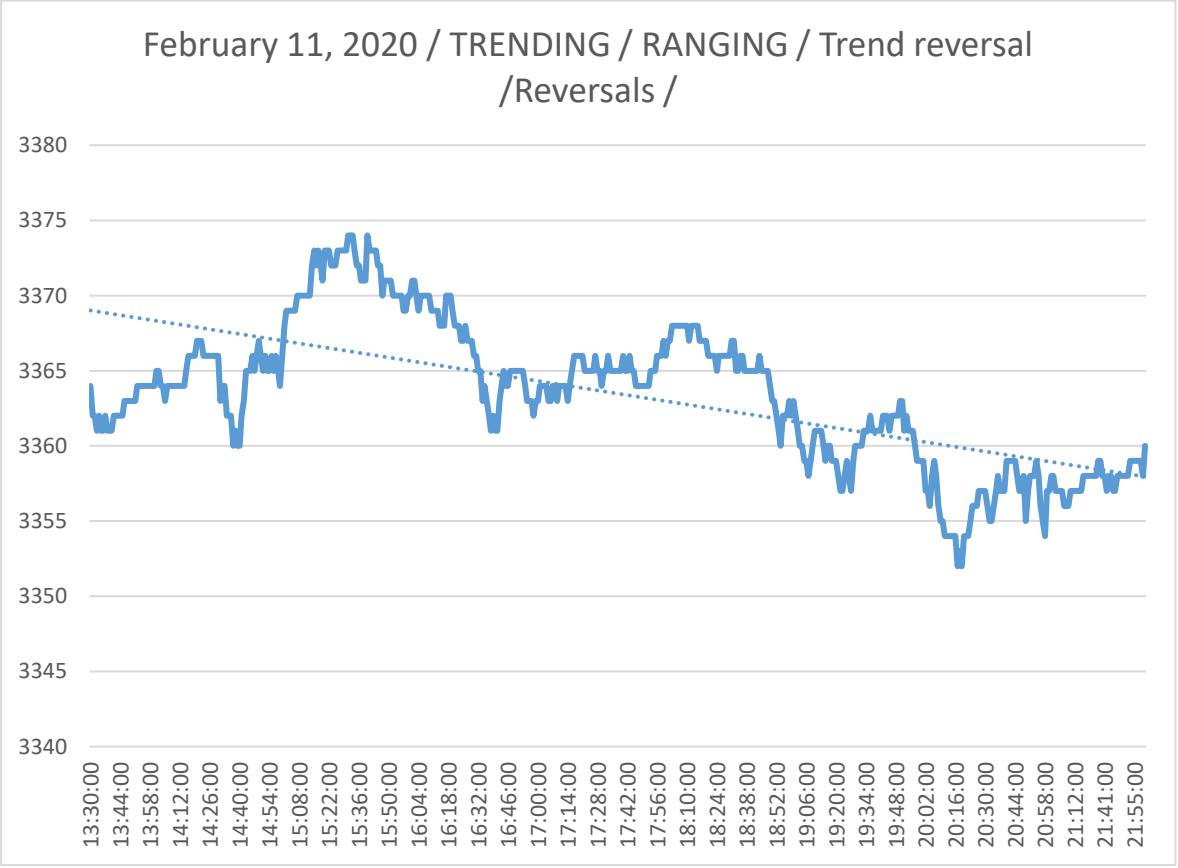

February 19, 2020 / TRENDING / Trend Reversal
3395
3390
3385
3380
3375
3370
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00

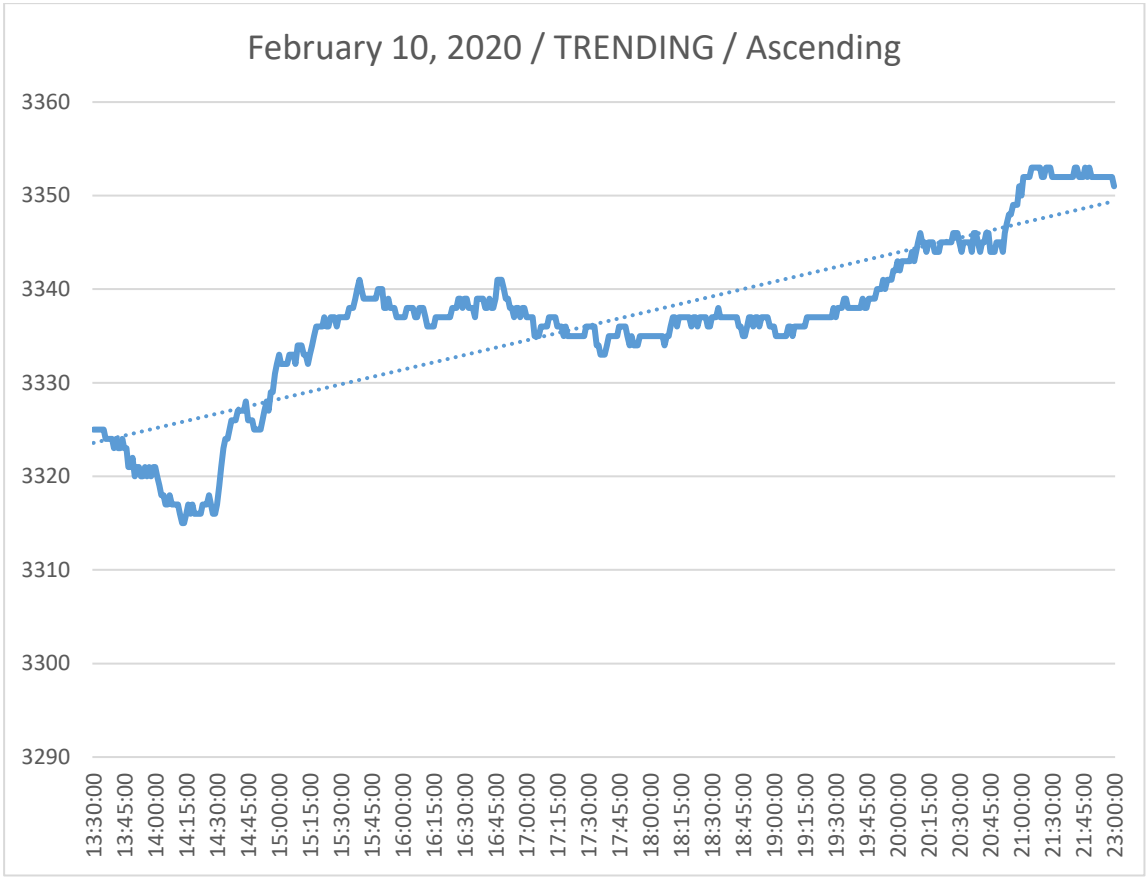

February 18, 2020 / RANGING / Random
3380
3375
3370
3365
3360
3355
3350
3345
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:03:00

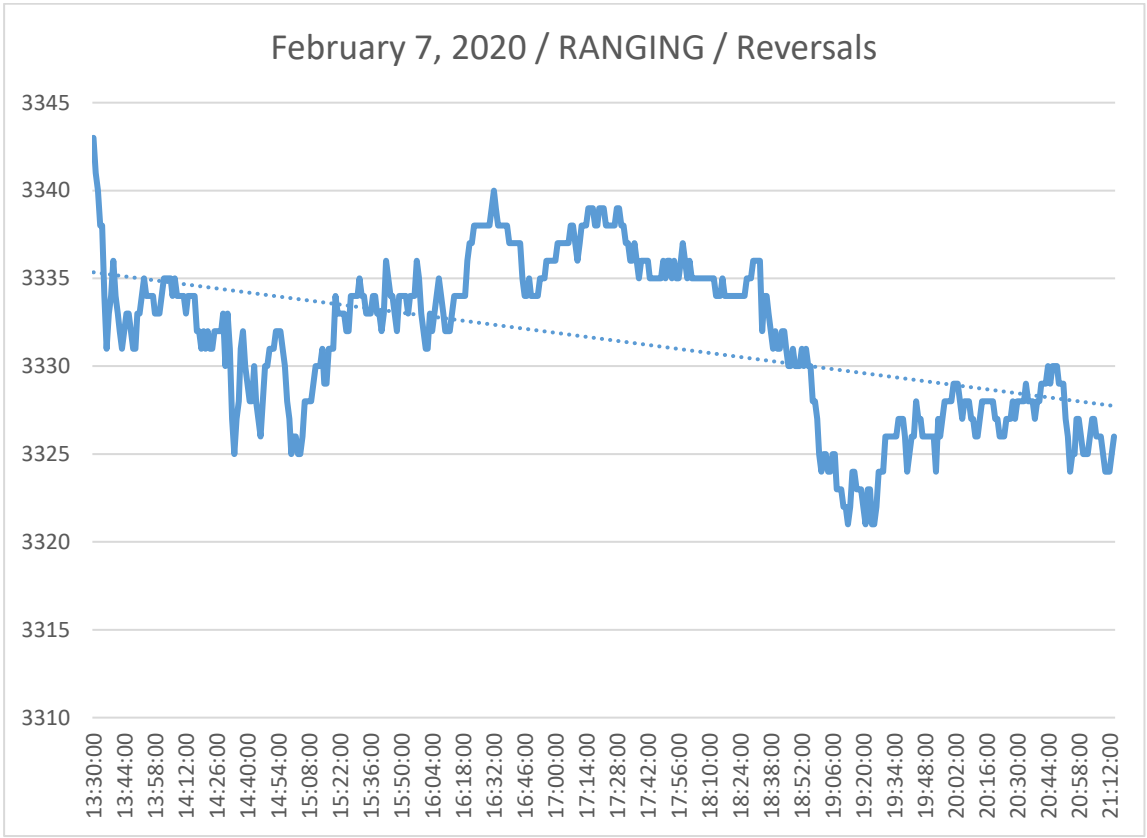

February 14, 2020 / TRENDING / RANGING
3385
3380
3375
3370
3365
3360
3355
13:30:00
13:44:00
13:58:00
14:12:00
14:26:00
14:40:00
14:54:00
15:08:00
15:22:00
15:36:00
15:50:00
16:04:00
16:18:00
16:32:00
16:46:00
17:00:00
17:14:00
17:28:00
17:42:00
17:56:00
18:10:00
18:24:00
18:38:00
18:52:00
19:06:00
19:20:00
19:34:00
19:48:00
20:02:00
20:16:00
20:30:00
20:44:00
20:58:00

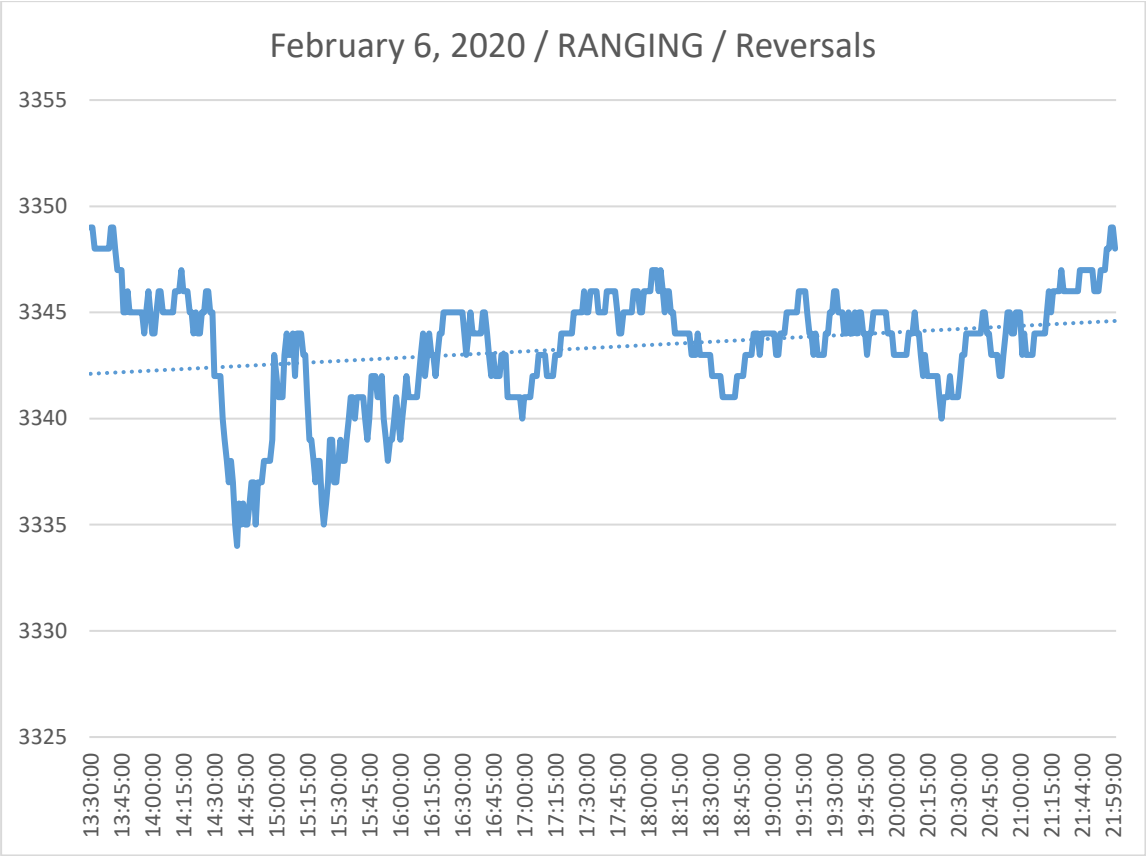

February 13, 2020 / TRENDING / Ascending / Trend Reversal
3385
3380
3375
3370
3365
3360
3355
3350
3345
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00

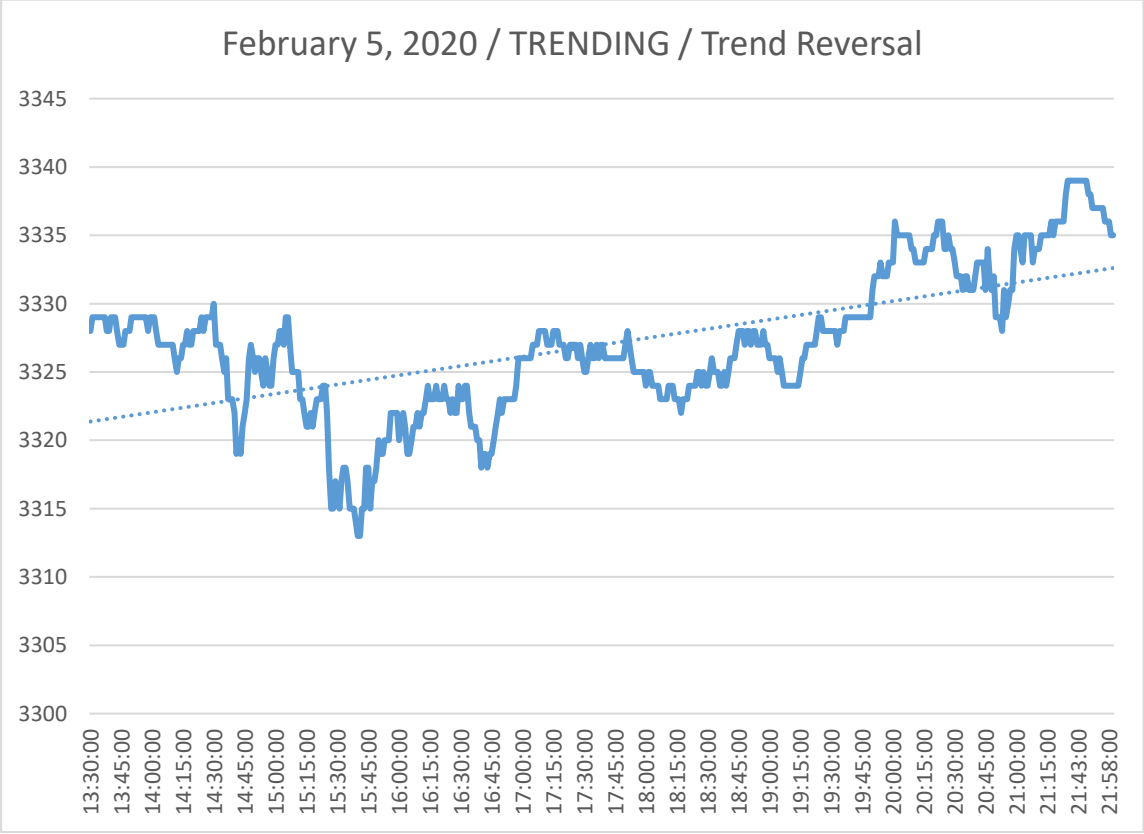

February 12, 2020 / TREDING / Ascending Trend
3385
3380
3375
3370
3365
3360
13:30:00
13:44:00
13:58:00
14:12:00
14:26:00
14:40:00
14:54:00
15:08:00
15:22:00
15:36:00
15:50:00
16:04:00
16:18:00
16:32:00
16:46:00
17:00:00
17:14:00
17:28:00
17:42:00
17:56:00
18:10:00
18:24:00
18:38:00
18:52:00
19:06:00
19:20:00
19:34:00
19:48:00
20:02:00
20:16:00
20:30:00
20:44:00
20:58:00
21:12:00
21:41:00
21:55:00

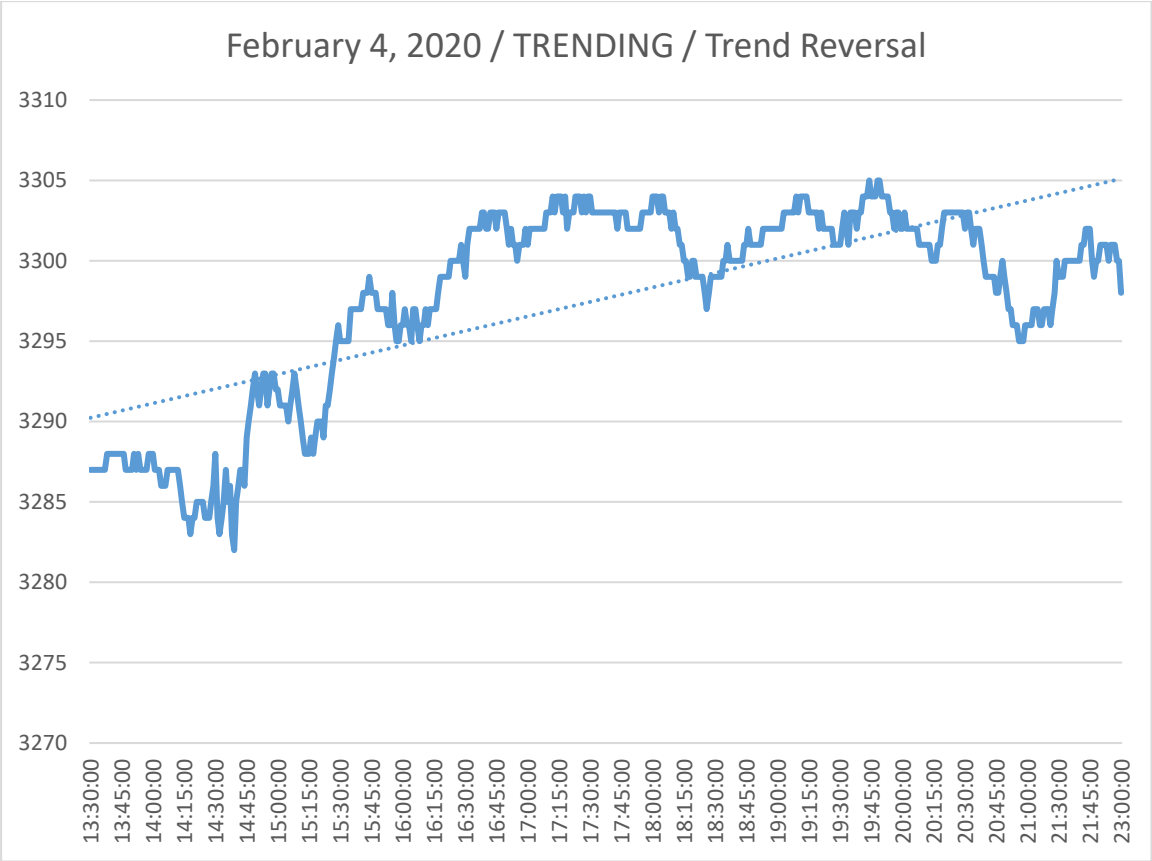

February 11, 2020 / TRENDING / RANGING / Trend reversal
/Reversals /
3380
3375
3370
3365
3360
3355
3350
3345
3340
13:30:00
13:44:00
13:58:00
14:12:00
14:26:00
14:40:00
14:54:00
15:08:00
15:22:00
15:36:00
15:50:00
16:04:00
16:18:00
16:32:00
16:46:00
17:00:00
17:14:00
17:28:00
17:42:00
17:56:00
18:10:00
18:24:00
18:38:00
18:52:00
19:06:00
19:20:00
19:34:00
19:48:00
20:02:00
20:16:00
20:30:00
20:44:00
20:58:00
21:12:00
21:41:00
21:55:00

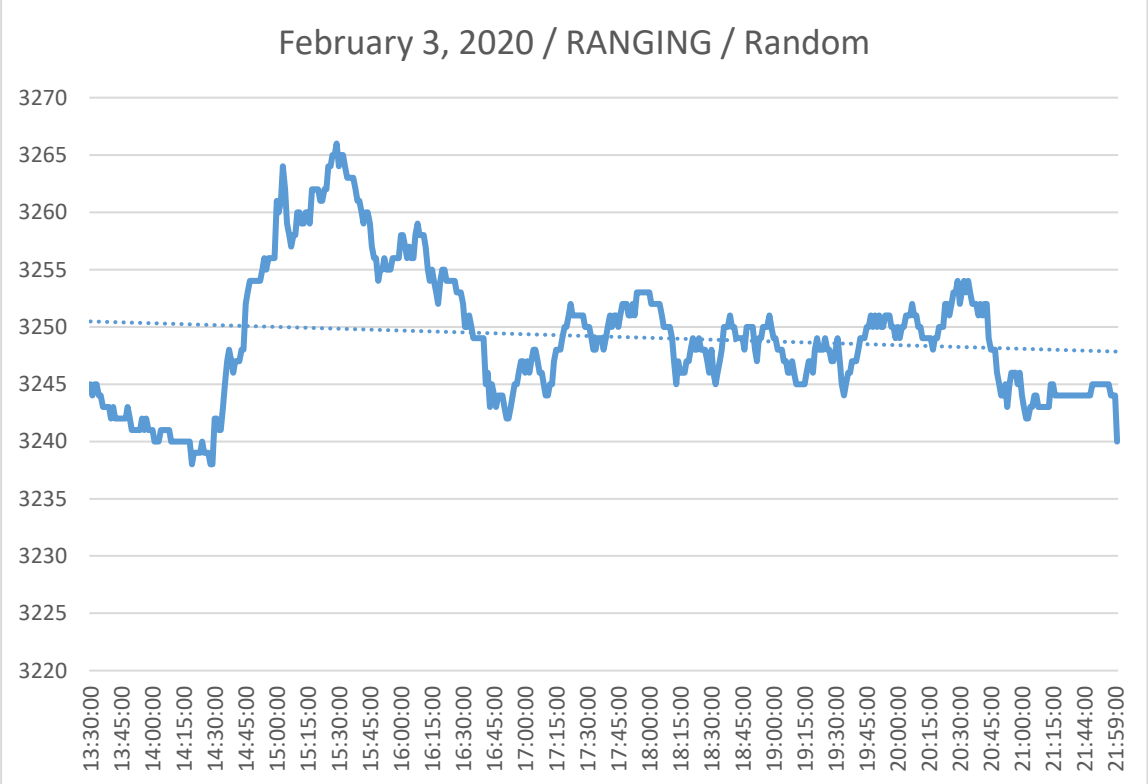

February 10, 2020 / TRENDING / Ascending
3360
3350
3340
3330
3320
3310
3300
3290
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00

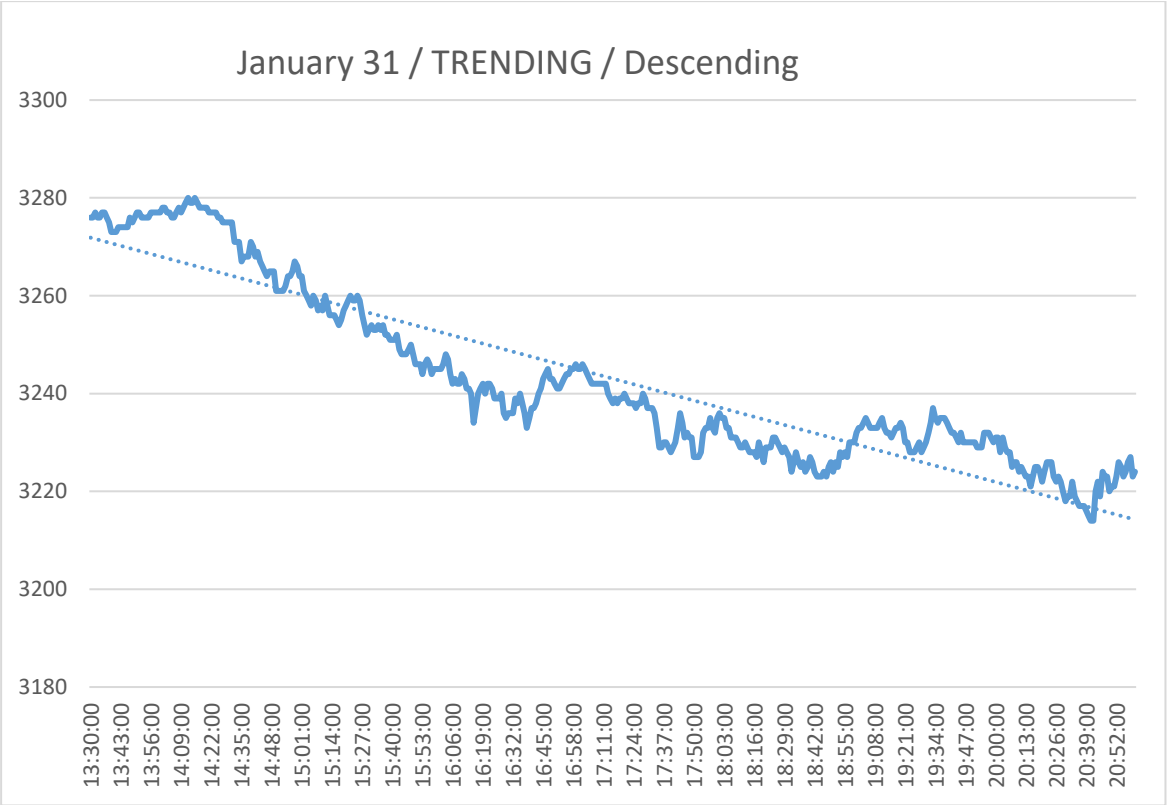

February 7, 2020 / RANGING / Reversals
3345
3340
3335
3330
3325
3320
3315
3310
13:30:00
13:44:00
13:58:00
14:12:00
14:26:00
14:40:00
14:54:00
15:08:00
15:22:00
15:36:00
15:50:00
16:04:00
16:18:00
16:32:00
16:46:00
17:00:00
17:14:00
17:28:00
17:42:00
17:56:00
18:10:00
18:24:00
18:38:00
18:52:00
19:06:00
19:20:00
19:34:00
19:48:00
20:02:00
20:16:00
20:30:00
20:44:00
20:58:00
21:12:00

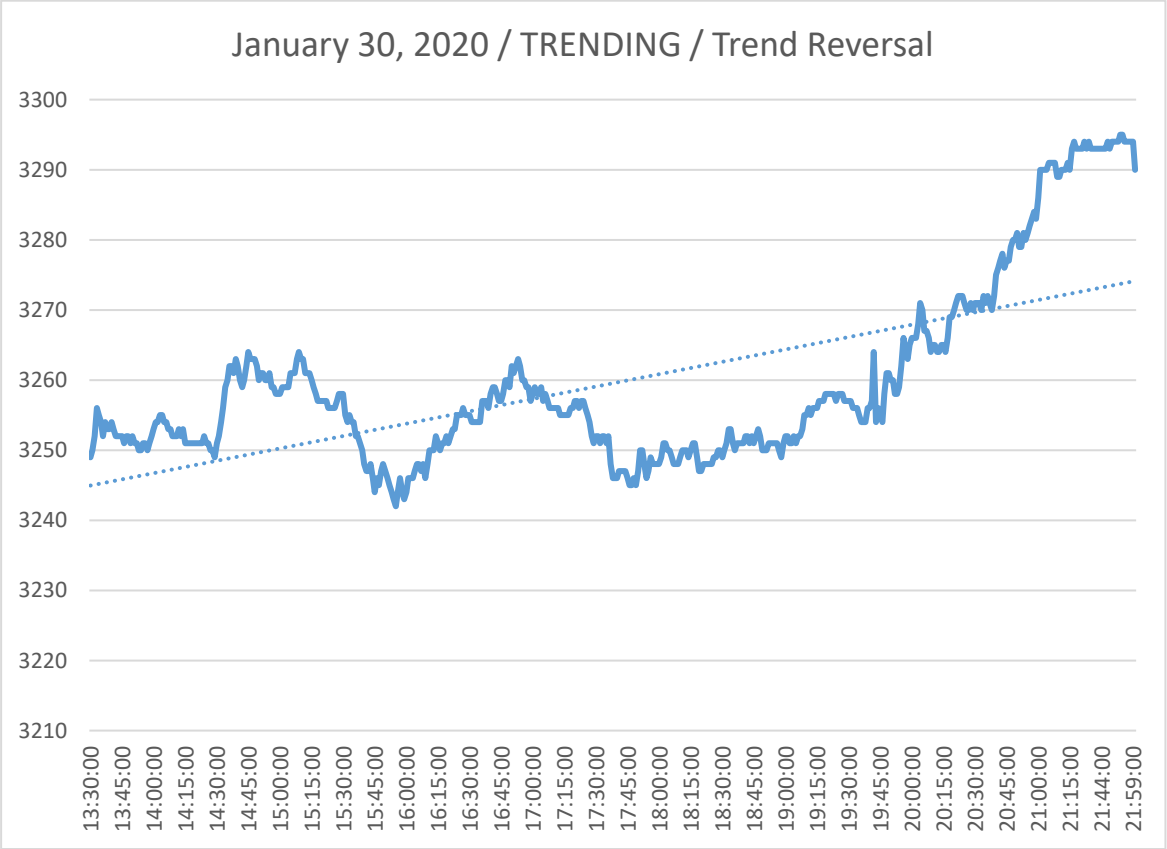

February 6, 2020 / RANGING / Reversals
3355
3350
3345
3340
3335
3330
3325
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:15:00
21:44:00
21:59:00

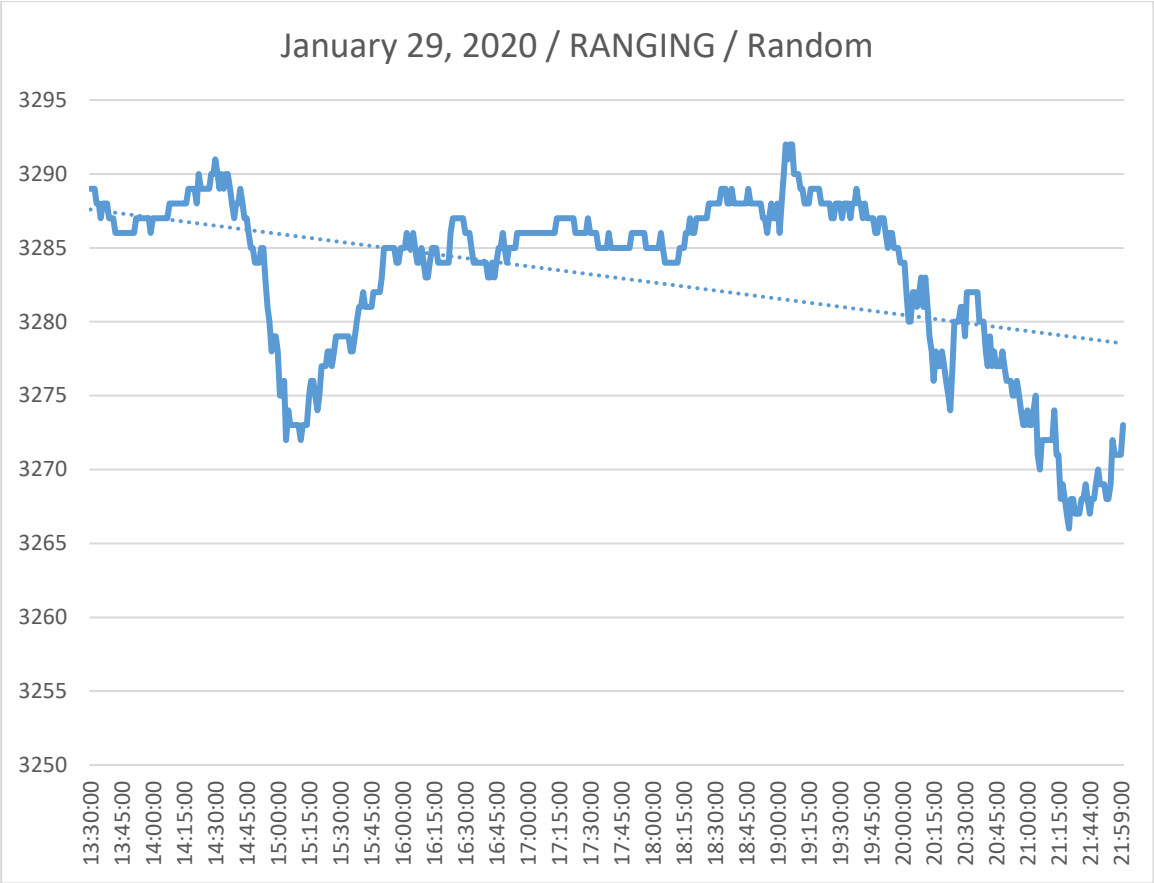

February 5, 2020 / TRENDING / Trend Reversal
3345
3340
3335
3330
3325
3320
3315
3310
3305
3300
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:15:00
21:43:00
21:58:00

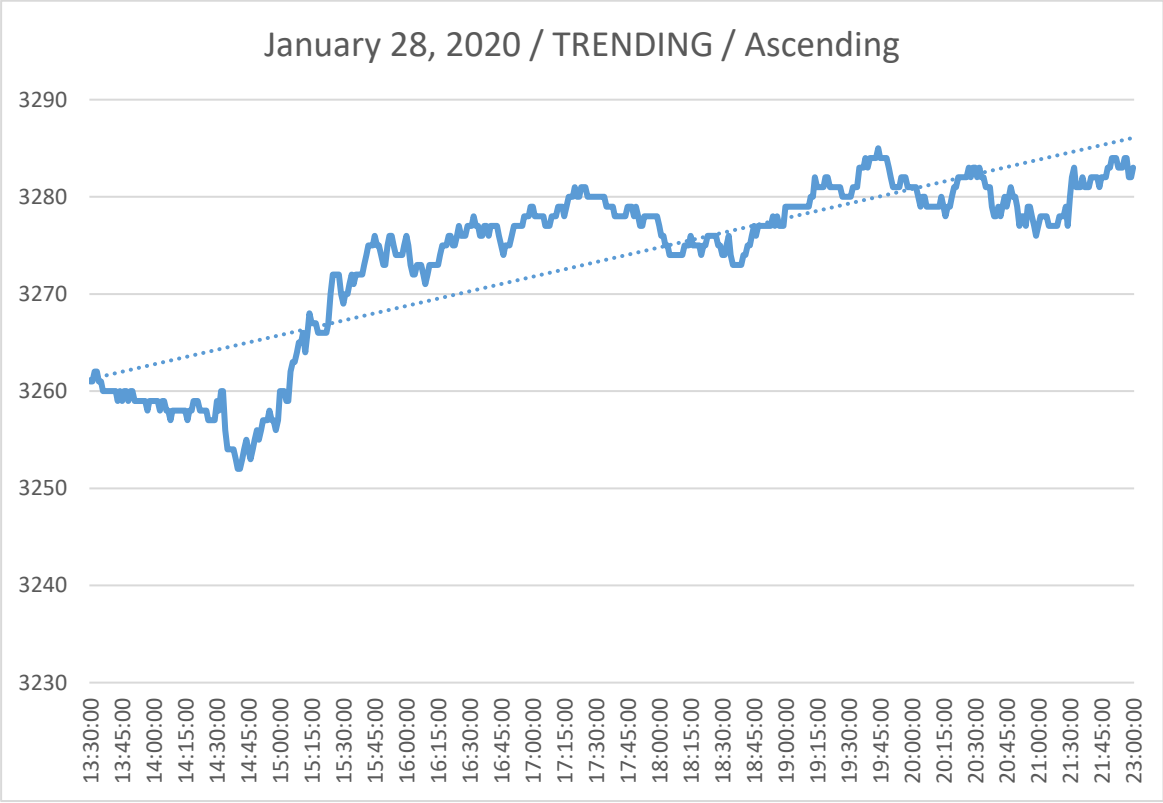

February 4, 2020 / TRENDING / Trend Reversal
3310
3305
3300
3295
3290
3285
3280
3275
3270
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00

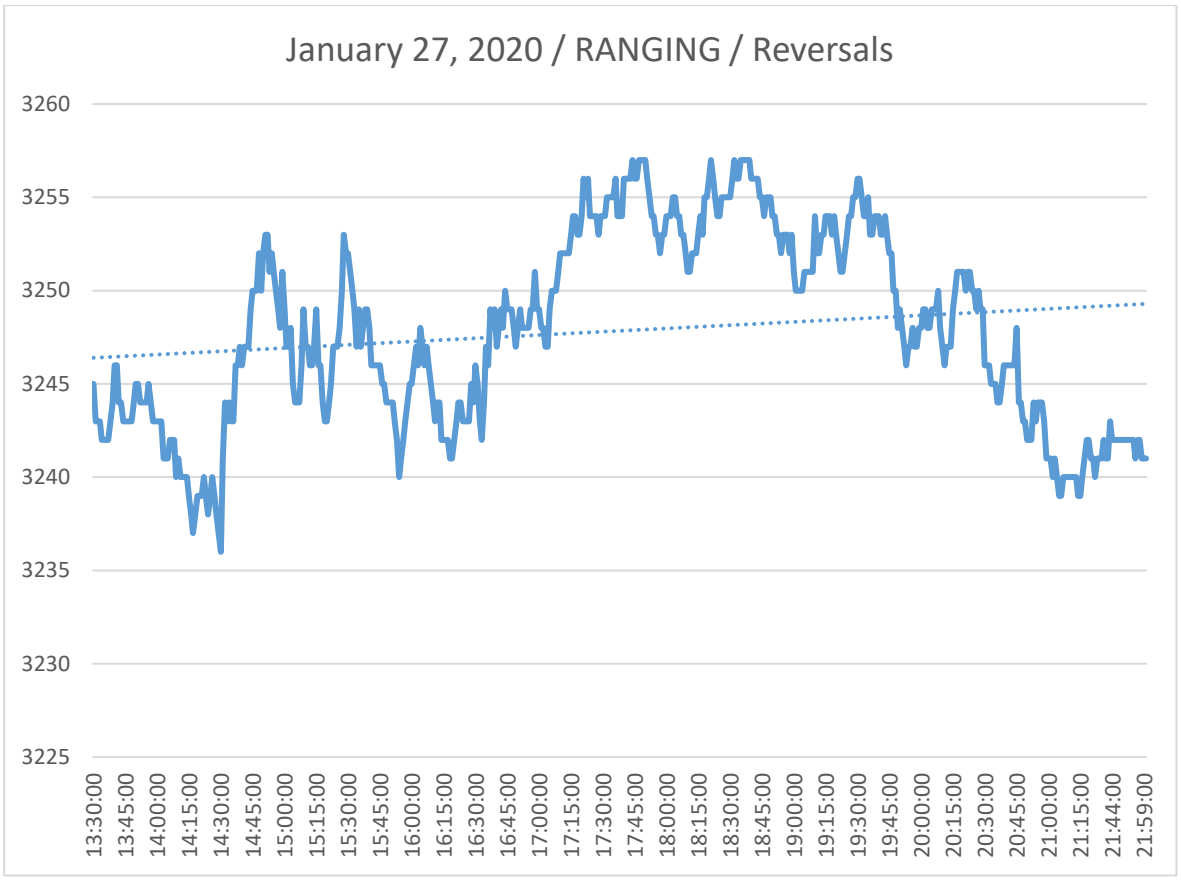

February 3, 2020 / RANGING / Random
3270
3265
3260
3255
3250
3245
3240
3235
3230
3225
3220
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:15:00
21:44:00
21:59:00

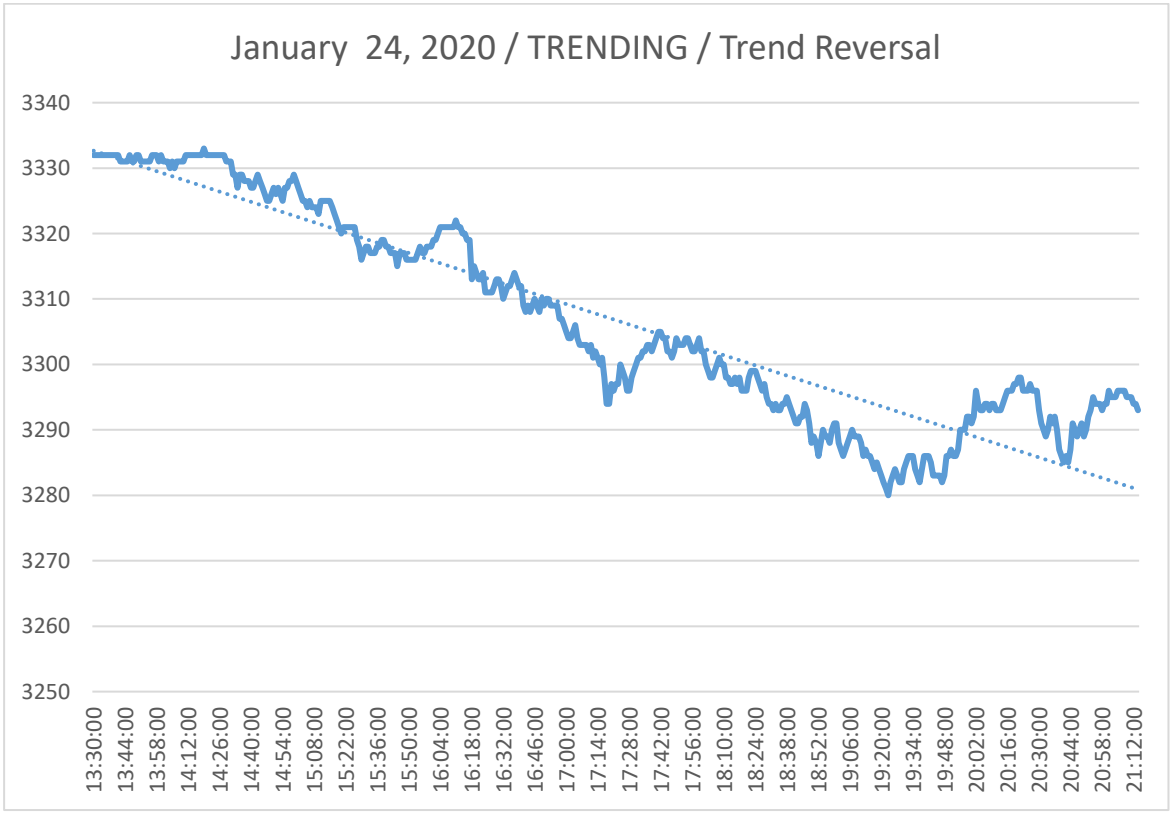

January 31 / TRENDING / Descending
3300
3280
3260
3240
3220
3200
3180
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:26:00
20:39:00
20:52:00

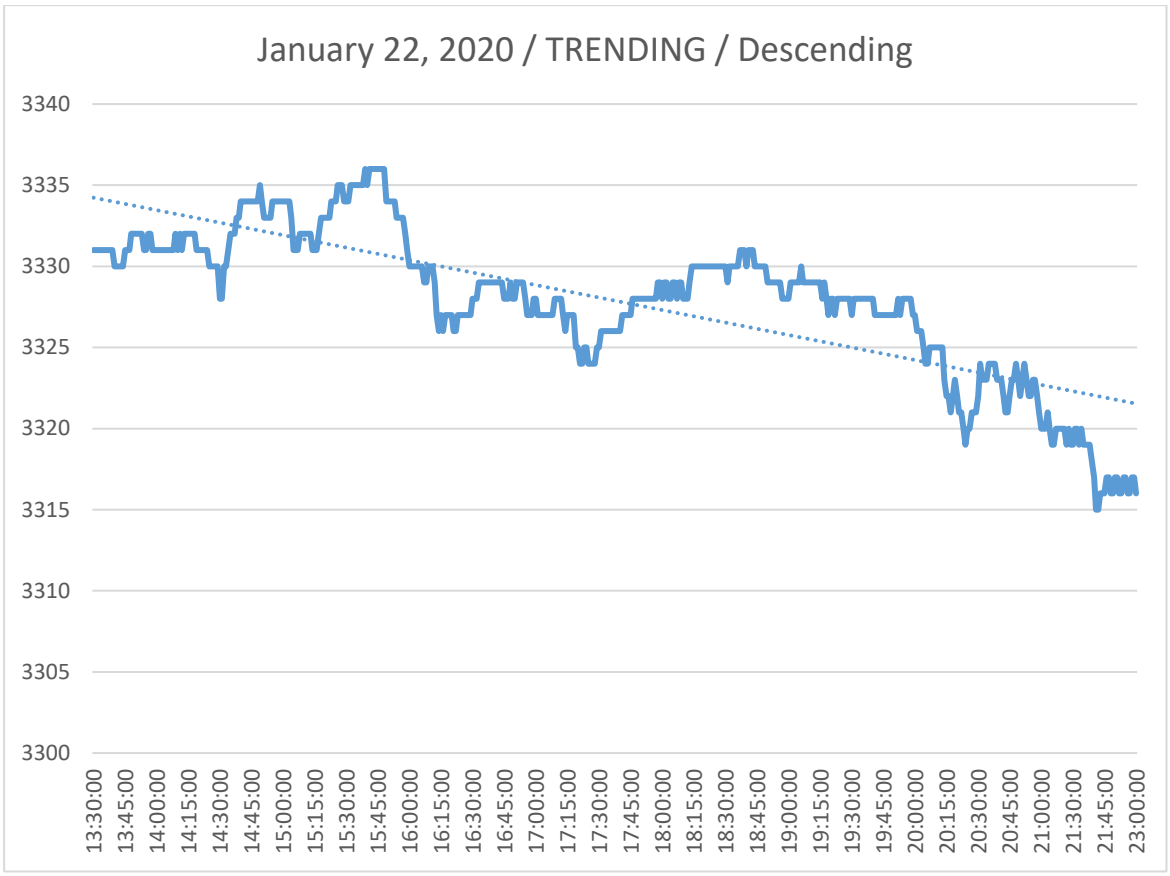

January 30, 2020 / TRENDING / Trend Reversal
3300
3290
3280
3270
3260
3250
3240
3230
3220
3210
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:15:00
21:44:00
21:59:00

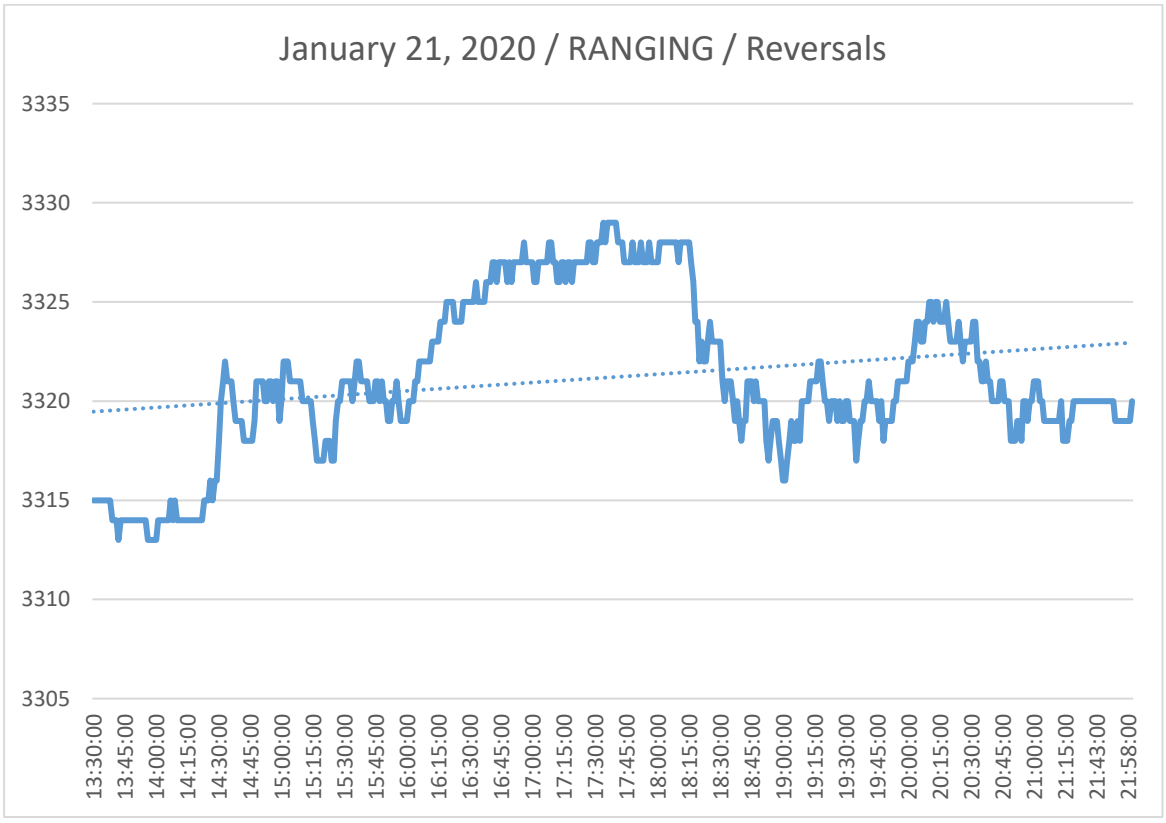

January 29, 2020 / RANGING / Random
3295
3290
3285
3280
3275
3270
3265
3260
3255
3250
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:15:00
21:44:00
21:59:00

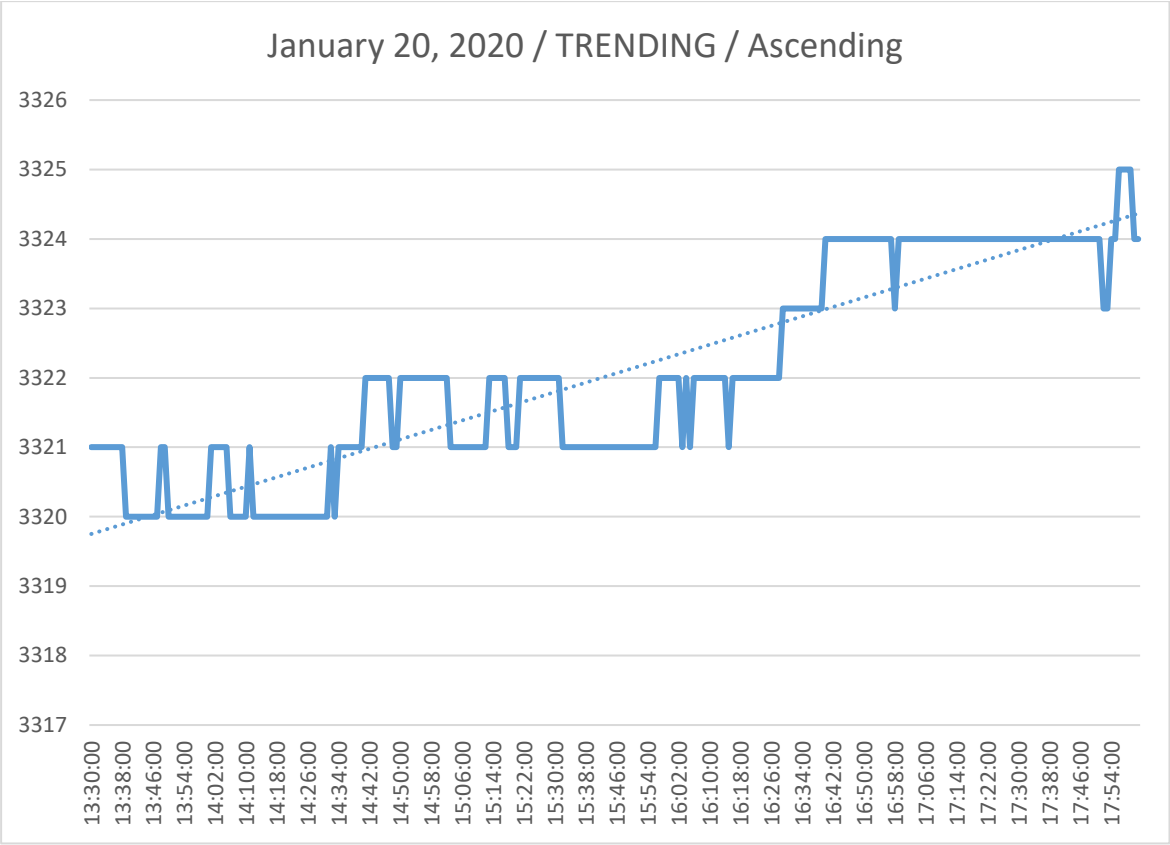

January 28, 2020 / TRENDING / Ascending
3290
3280
3270
3260
3250
3240
3230
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00

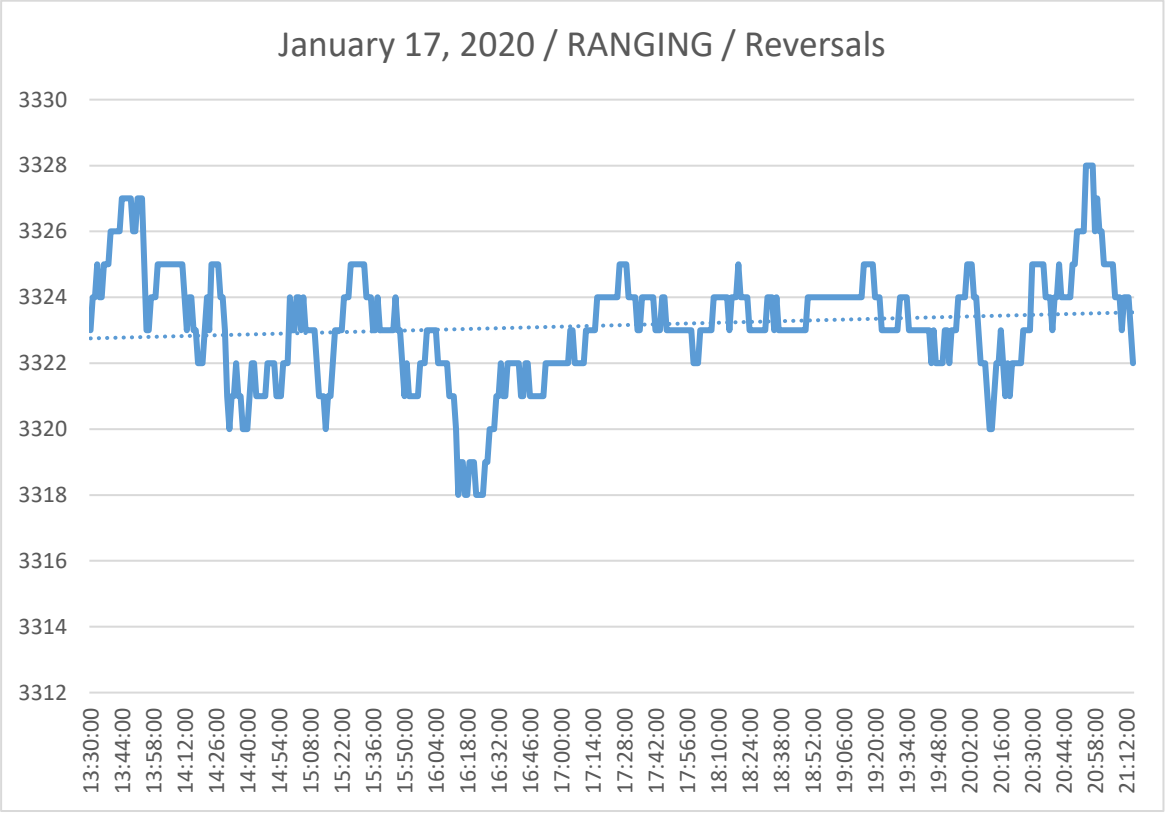

January 27, 2020 / RANGING / Reversals
3260
3255
3250
3245
3240
3235
3230
3225
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:15:00
21:44:00
21:59:00

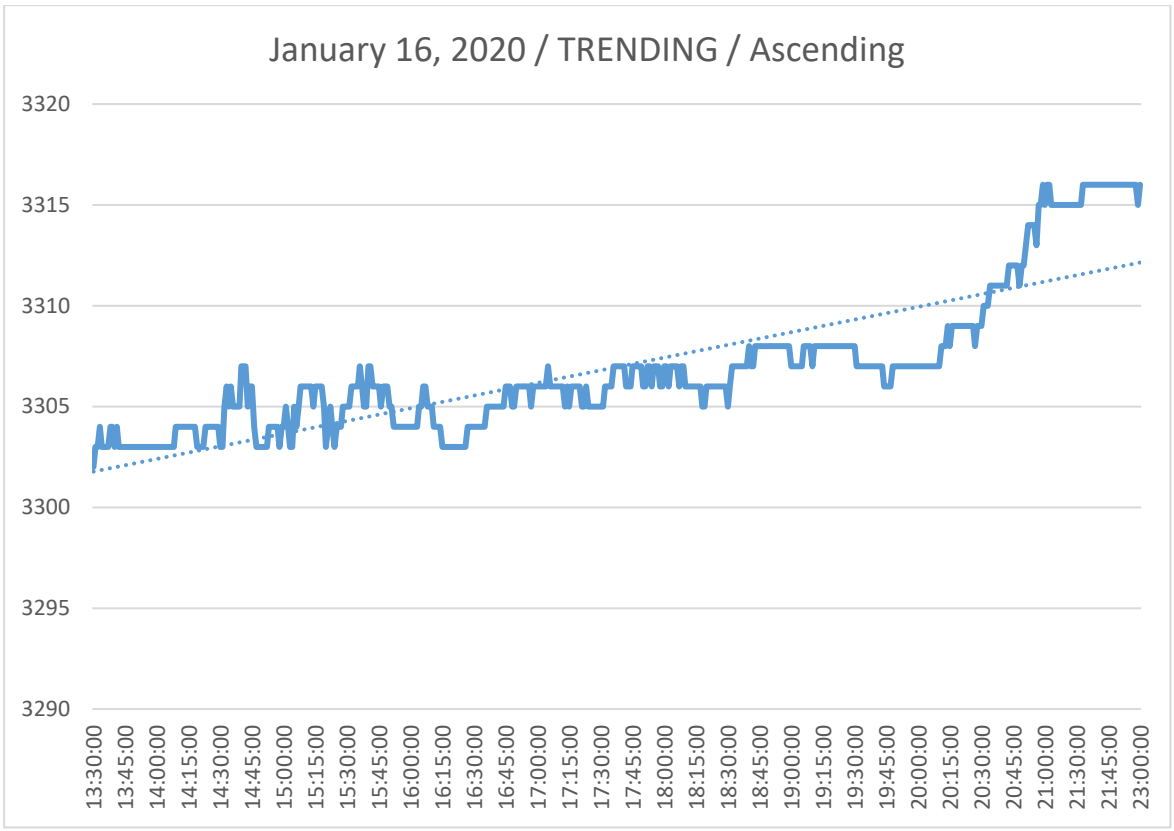

January 24, 2020 / TRENDING / Trend Reversal
3340
3330
3320
3310
3300
3290
3280
3270
3260
3250
13:30:00
13:44:00
13:58:00
14:12:00
14:26:00
14:40:00
14:54:00
15:08:00
15:22:00
15:36:00
15:50:00
16:04:00
16:18:00
16:32:00
16:46:00
17:00:00
17:14:00
17:28:00
17:42:00
17:56:00
18:10:00
18:24:00
18:38:00
18:52:00
19:06:00
19:20:00
19:34:00
19:48:00
20:02:00
20:16:00
20:30:00
20:44:00
20:58:00
21:12:00

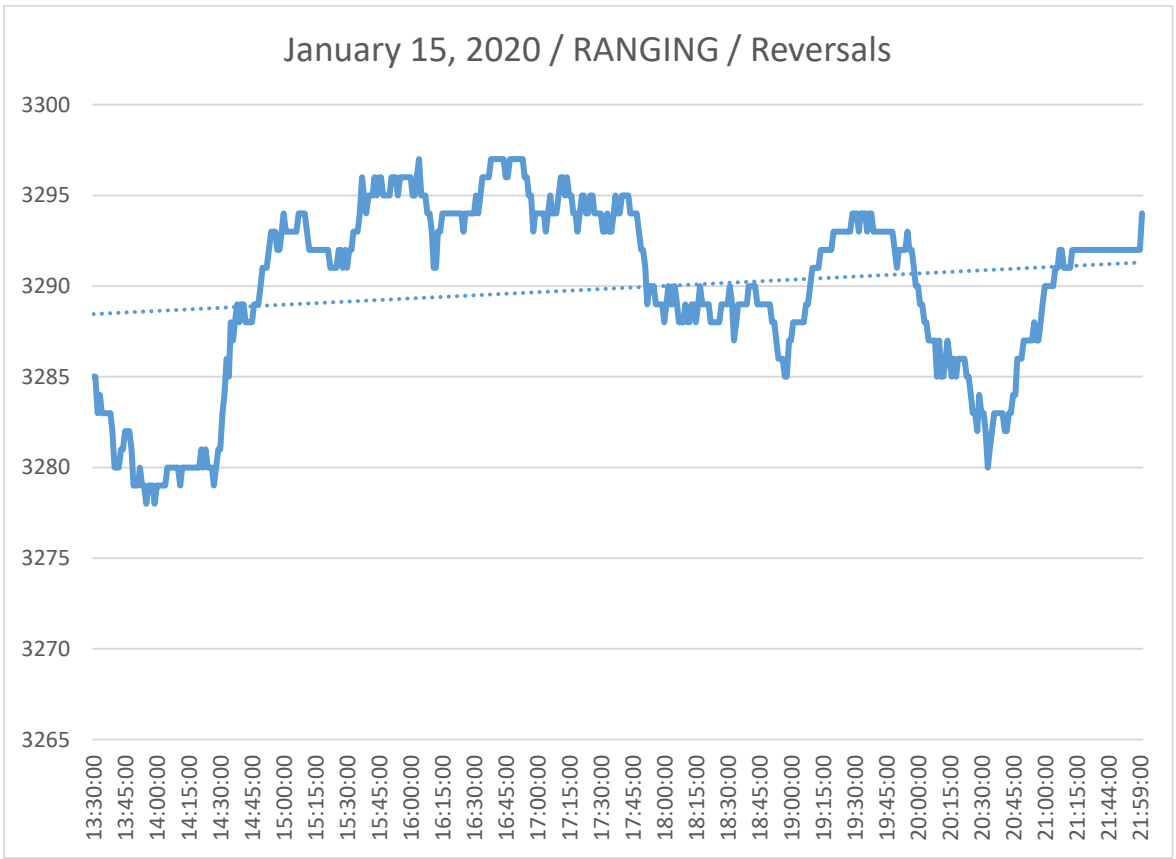

January 22, 2020 / TRENDING / Descending
3340
3335
3330
3325
3320
3315
3310
3305
3300
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00

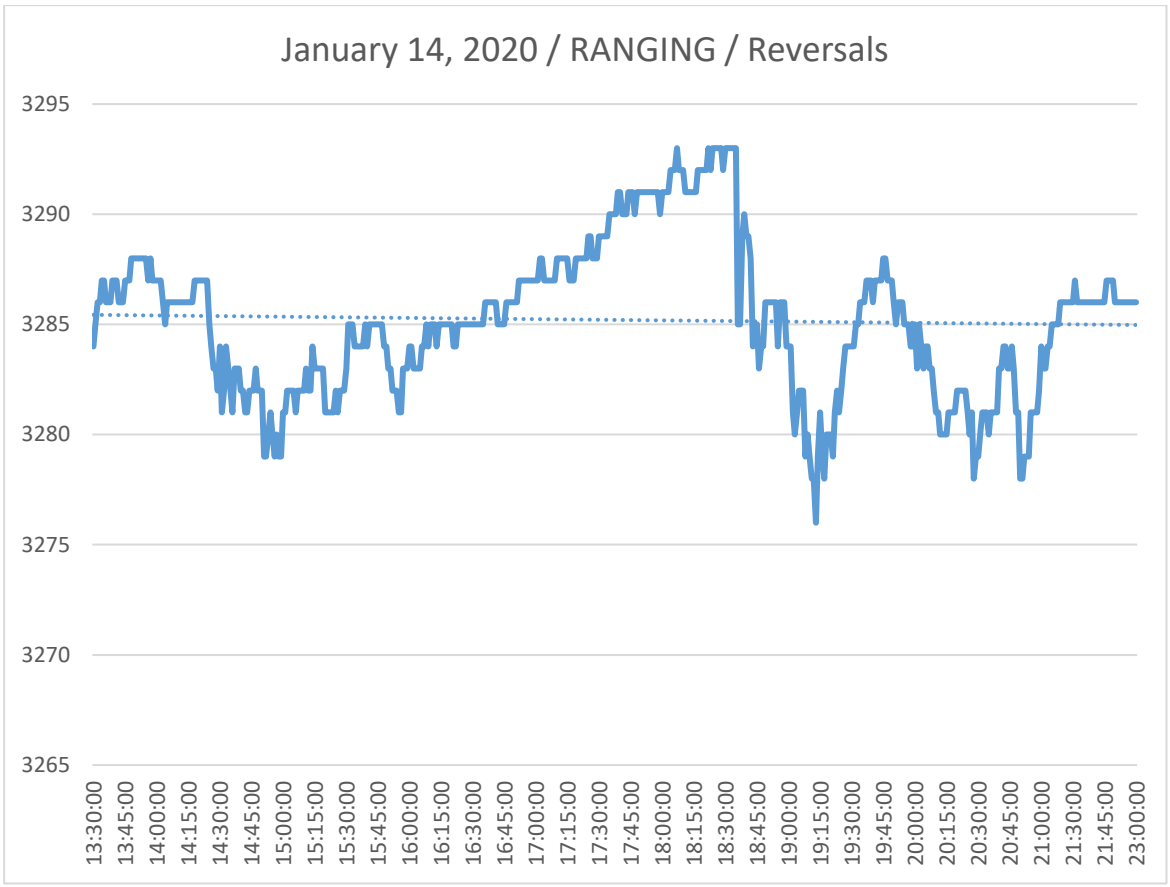

January 21, 2020 / RANGING / Reversals
3335
3330
3325
3320
3315
3310
3305
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:15:00
21:43:00
21:58:00

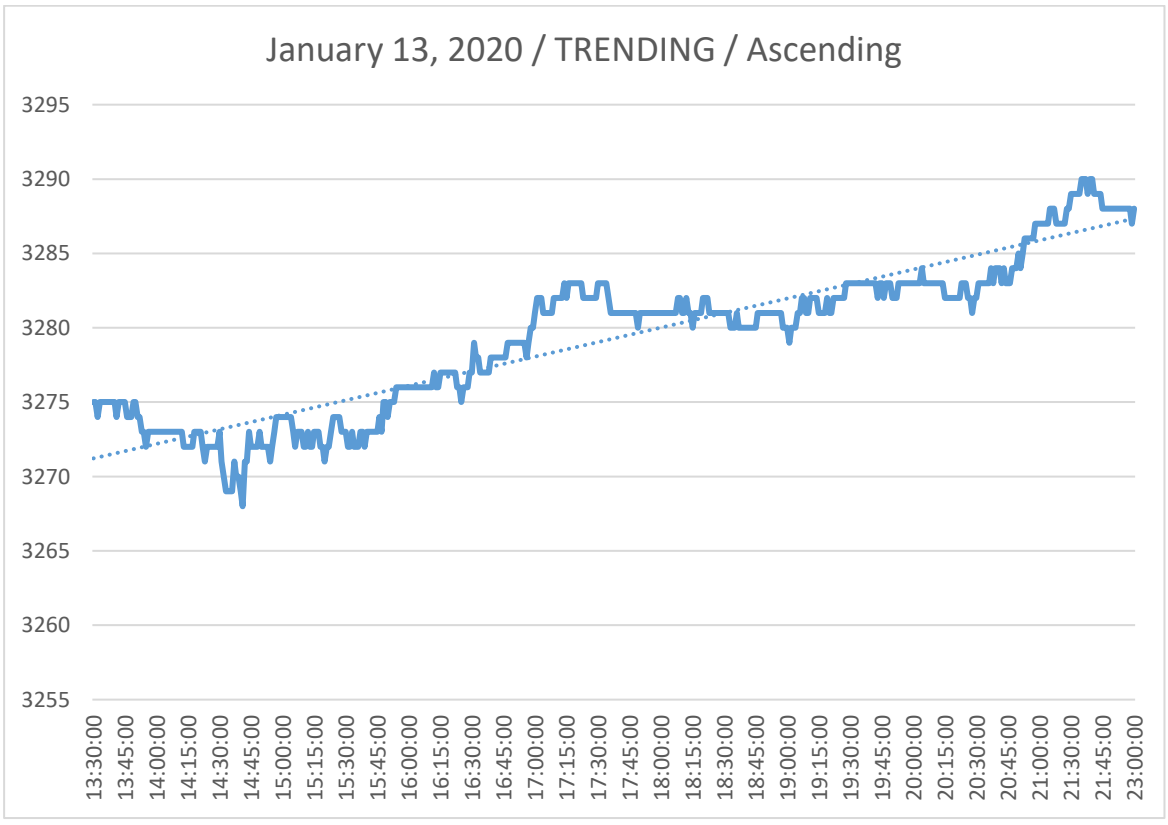

January 20, 2020 / TRENDING / Ascending
3326
3325
3324
3323
3322
3321
3320
3319
3318
3317
13:30:00
13:38:00
13:46:00
13:54:00
14:02:00
14:10:00
14:18:00
14:26:00
14:34:00
14:42:00
14:50:00
14:58:00
15:06:00
15:14:00
15:22:00
15:30:00
15:38:00
15:46:00
15:54:00
16:02:00
16:10:00
16:18:00
16:26:00
16:34:00
16:42:00
16:50:00
16:58:00
17:06:00
17:14:00
17:22:00
17:30:00
17:38:00
17:46:00
17:54:00

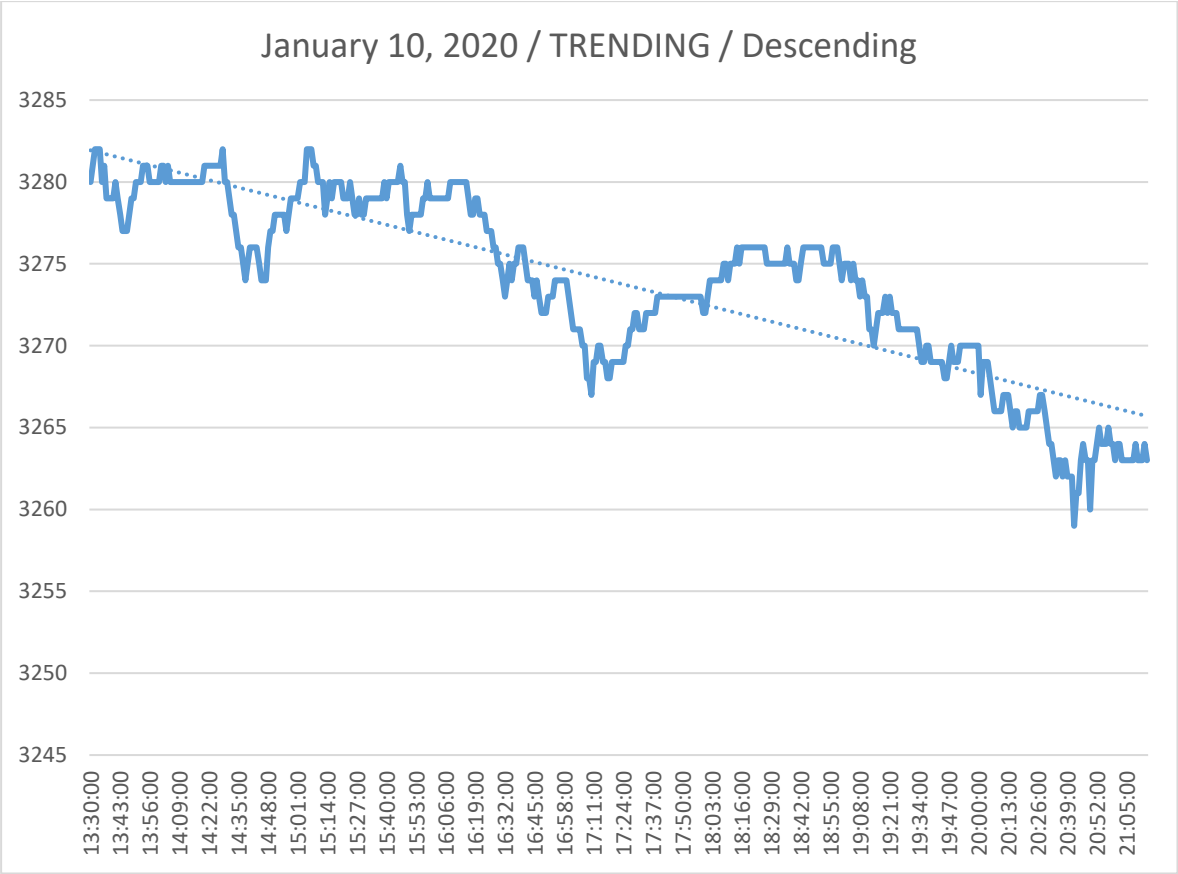

January 17, 2020 / RANGING / Reversals
3330
3328
3326
3324
3322
3320
3318
3316
3314
3312
13:30:00
13:44:00
13:58:00
14:12:00
14:26:00
14:40:00
14:54:00
15:08:00
15:22:00
15:36:00
15:50:00
16:04:00
16:18:00
16:32:00
16:46:00
17:00:00
17:14:00
17:28:00
17:42:00
17:56:00
18:10:00
18:24:00
18:38:00
18:52:00
19:06:00
19:20:00
19:34:00
19:48:00
20:02:00
20:16:00
20:30:00
20:44:00
20:58:00
21:12:00

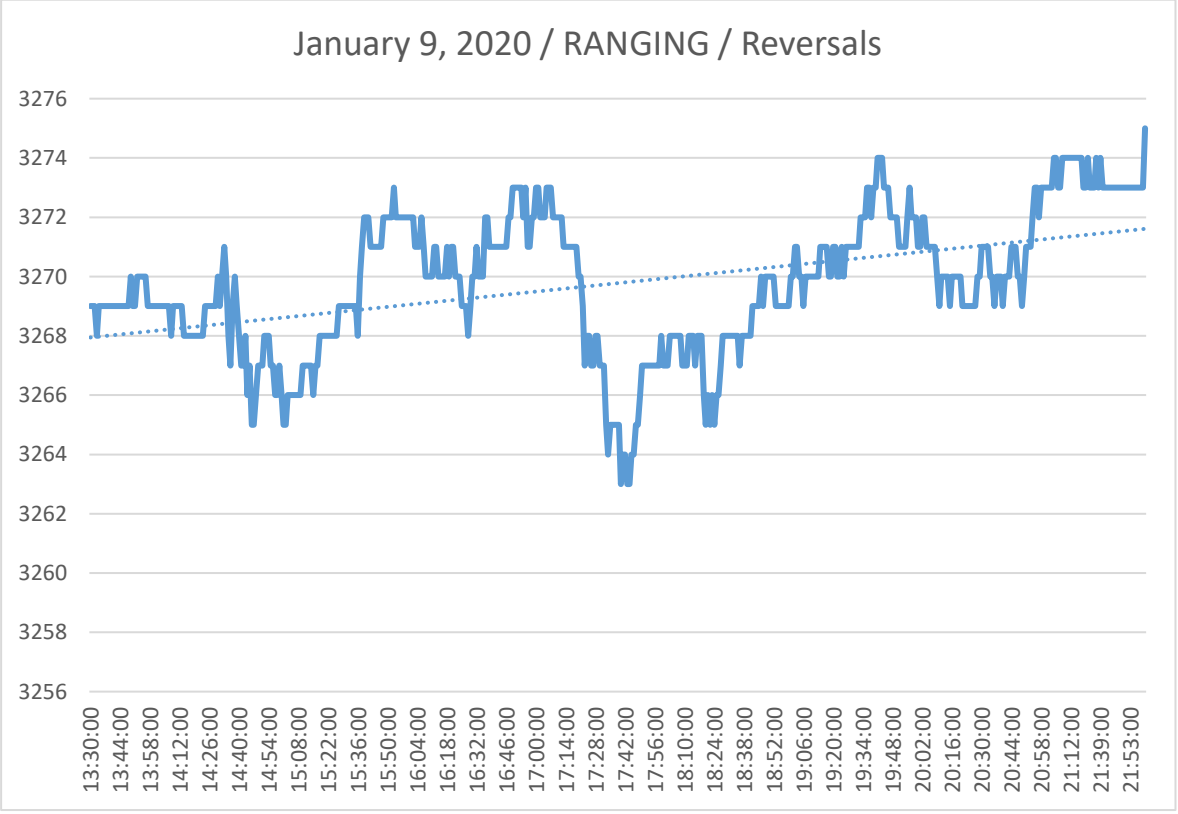

January 16, 2020 / TRENDING / Ascending
3320
3315
3310
3305
3300
3295
3290
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00

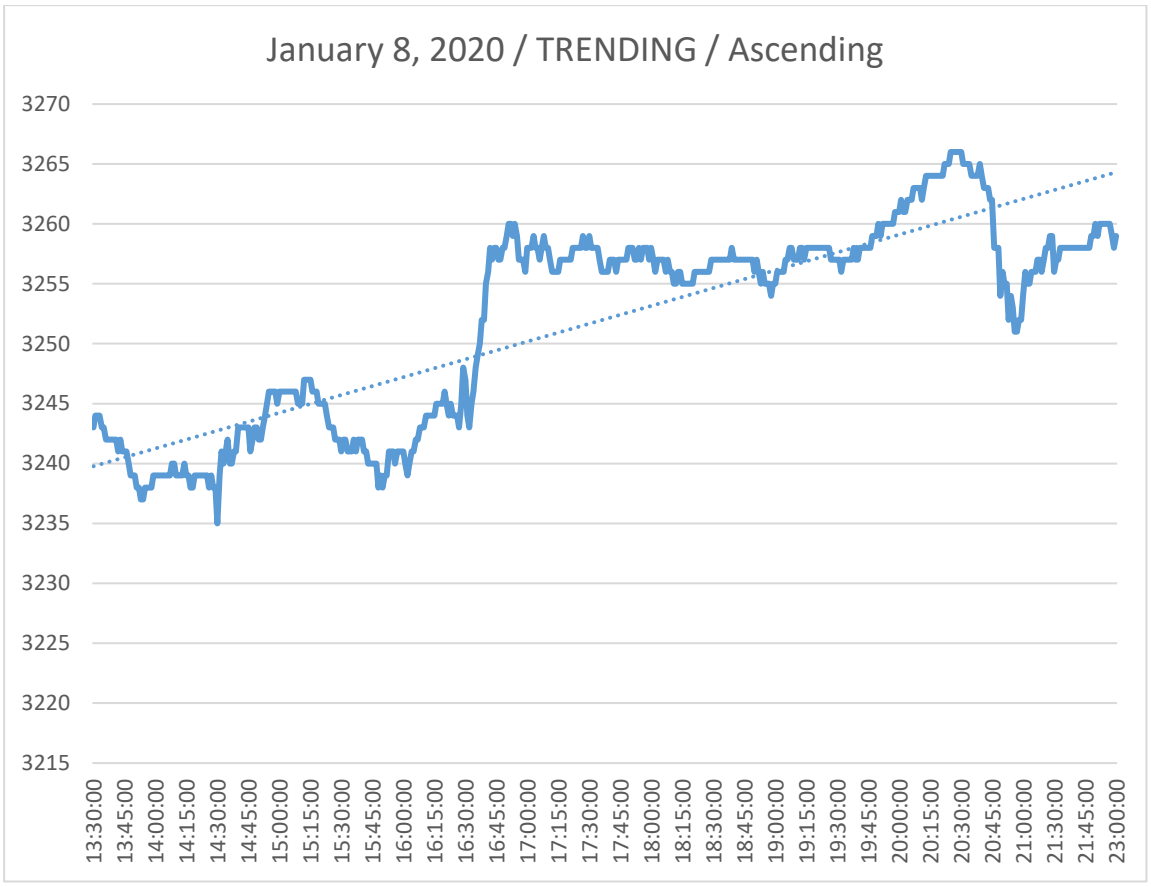

January 15, 2020 / RANGING / Reversals
3300
3295
3290
3285
3280
3275
3270
3265
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:15:00
21:44:00
21:59:00

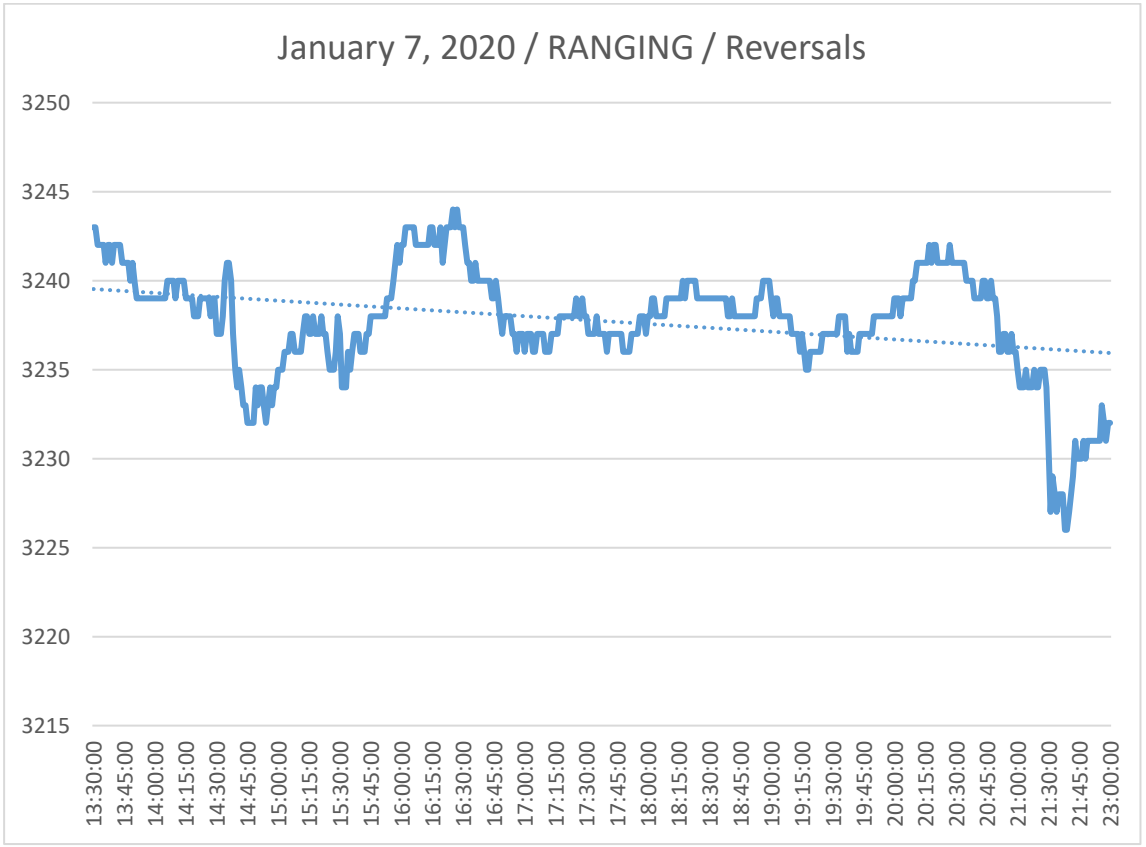

January 14, 2020 / RANGING / Reversals
3295
3290
3285
3280
3275
3270
3265
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00

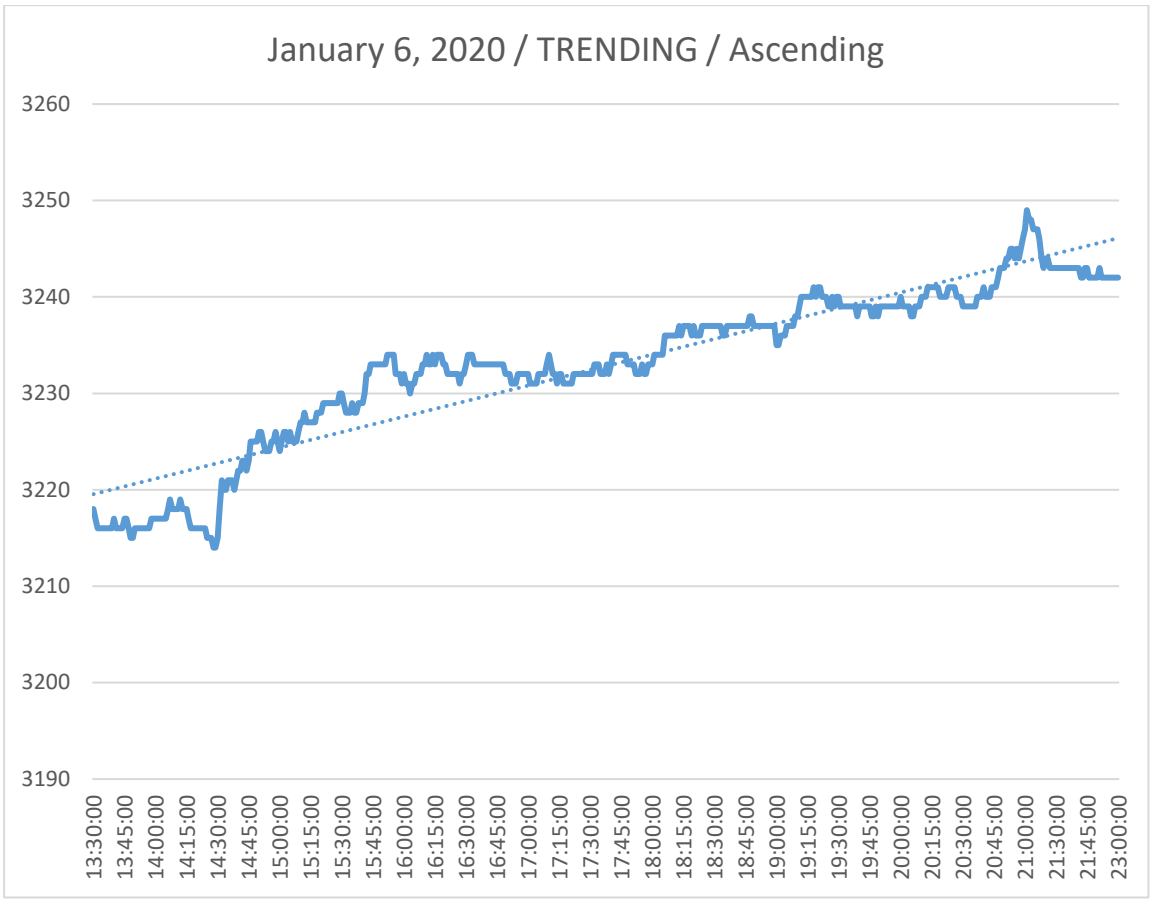

January 13, 2020 / TRENDING / Ascending
3295
3290
3285
3280
3275
3270
3265
3260
3255
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00

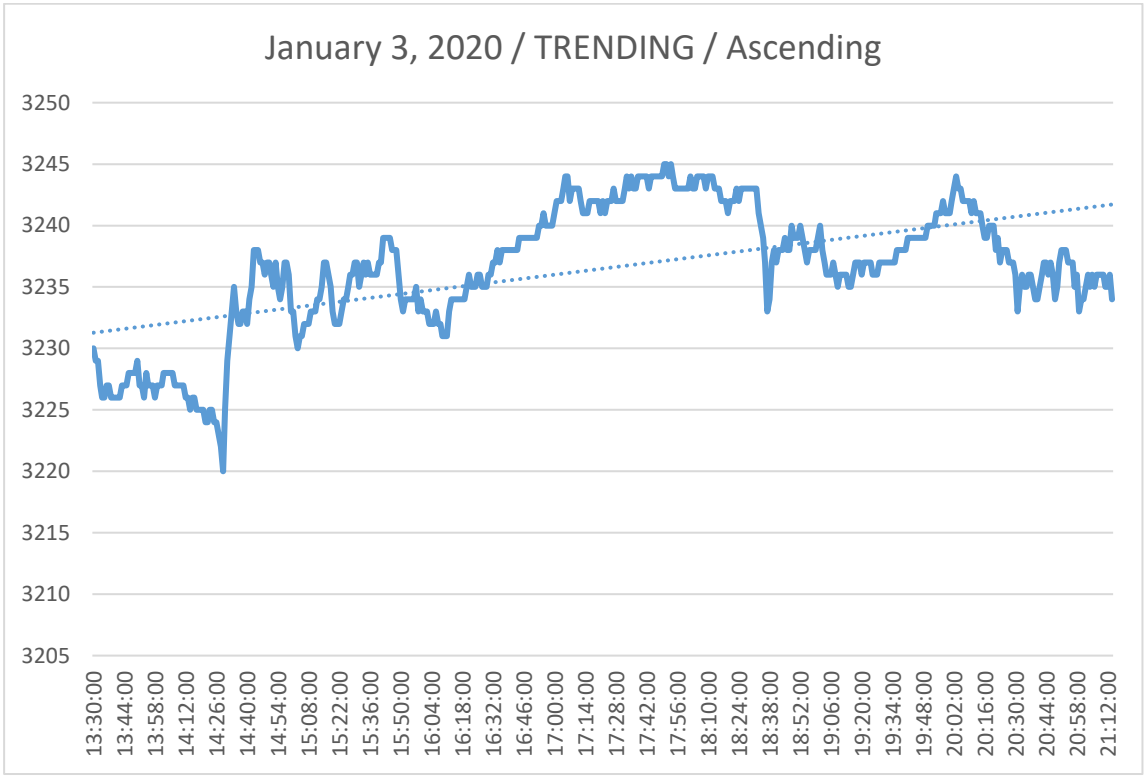

January 10, 2020 / TRENDING / Descending
3285
3280
3275
3270
3265
3260
3255
3250
3245
13:30:00
13:43:00
13:56:00
14:09:00
14:22:00
14:35:00
14:48:00
15:01:00
15:14:00
15:27:00
15:40:00
15:53:00
16:06:00
16:19:00
16:32:00
16:45:00
16:58:00
17:11:00
17:24:00
17:37:00
17:50:00
18:03:00
18:16:00
18:29:00
18:42:00
18:55:00
19:08:00
19:21:00
19:34:00
19:47:00
20:00:00
20:13:00
20:26:00
20:39:00
20:52:00
21:05:00

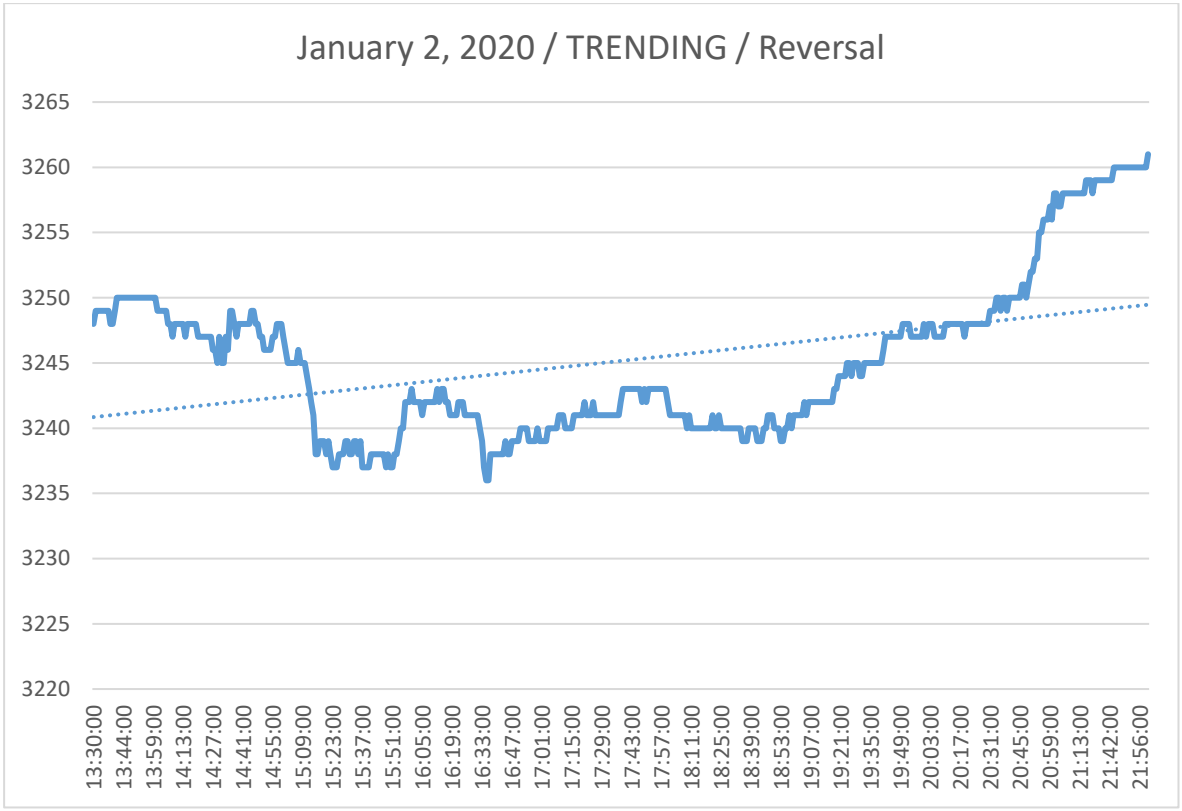

January 9, 2020 / RANGING / Reversals
3276
3274
3272
3270
3268
3266
3264
3262
3260
3258
3256
13:30:00
13:44:00
13:58:00
14:12:00
14:26:00
14:40:00
14:54:00
15:08:00
15:22:00
15:36:00
15:50:00
16:04:00
16:18:00
16:32:00
16:46:00
17:00:00
17:14:00
17:28:00
17:42:00
17:56:00
18:10:00
18:24:00
18:38:00
18:52:00
19:06:00
19:20:00
19:34:00
19:48:00
20:02:00
20:16:00
20:30:00
20:44:00
20:58:00
21:12:00
21:39:00
21:53:00


January 8, 2020 / TRENDING / Ascending
3270
3265
3260
3255
3250
3245
3240
3235
3230
3225
3220
3215
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00


January 7, 2020 / RANGING / Reversals
3250
3245
3240
3235
3230
3225
3220
3215
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00


January 6, 2020 / TRENDING / Ascending
3260
3250
3240
3230
3220
3210
3200
3190
13:30:00
13:45:00
14:00:00
14:15:00
14:30:00
14:45:00
15:00:00
15:15:00
15:30:00
15:45:00
16:00:00
16:15:00
16:30:00
16:45:00
17:00:00
17:15:00
17:30:00
17:45:00
18:00:00
18:15:00
18:30:00
18:45:00
19:00:00
19:15:00
19:30:00
19:45:00
20:00:00
20:15:00
20:30:00
20:45:00
21:00:00
21:30:00
21:45:00
23:00:00


January 3, 2020 / TRENDING / Ascending
3250
3245
3240
3235
3230
3225
3220
3215
3210
3205
13:30:00
13:44:00
13:58:00
14:12:00
14:26:00
14:40:00
14:54:00
15:08:00
15:22:00
15:36:00
15:50:00
16:04:00
16:18:00
16:32:00
16:46:00
17:00:00
17:14:00
17:28:00
17:42:00
17:56:00
18:10:00
18:24:00
18:38:00
18:52:00
19:06:00
19:20:00
19:34:00
19:48:00
20:02:00
20:16:00
20:30:00
20:44:00
20:58:00
21:12:00


January 2, 2020 / TRENDING / Reversal
3265
3260
3255
3250
3245
3240
3235
3230
3225
3220
13:30:00
13:44:00
13:59:00
14:13:00
14:27:00
14:41:00
14:55:00
15:09:00
15:23:00
15:37:00
15:51:00
16:05:00
16:19:00
16:33:00
16:47:00
17:01:00
17:15:00
17:29:00
17:43:00
17:57:00
18:11:00
18:25:00
18:39:00
18:53:00
19:07:00
19:21:00
19:35:00
19:49:00
20:03:00
20:17:00
20:31:00
20:45:00
20:59:00
21:13:00
21:42:00
21:56:00