Pertanika J. Sci. & Technol. 26 (1): 441 - 462 (2018)

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Parallel Exponential Smoothing Using the Bootstrap Method in R for Forecasting Asteroid's Orbital Elements

Lala Septem Riza[1]*, Judhistira Aria Utama[2], Syandi Mufti Putra[1], Ferry Mukharradi Simatupang[3] and Eddy Prasetyo Nugroho[1]

[1]Department of Computer Science Education, Universitas Pendidikan Indonesia, Jl. Setiabudhi, Bandung, Indonesia
[2]Department of Physics Education, Universitas Pendidikan Indonesia, Jl. Setiabudhi, Bandung, Indonesia
[3]Astronomy Research Division, Institut Teknologi Bandung, Jl. Ganesha, Bandung, Indonesia

ABSTRACT

Nowadays, large datasets become main intentions of researchers in many areas. However, a challenge that still remains mainly unresolved is the lack of strategies used for analysing large time-series datasets in parallel. Therefore, this research aims to design a model of exponential smoothing working on parallel computing by using the bootstrap method. Three parts will be considered in the model: data pre-processing using the bootstrap methods, parallel exponential smoothing, and aggregation of results to be the final predicted values. To implement the processes, some packages available in the R environment such as "foreach", "forecast" and "doParallel" are utilised. R environment provides many packages for scientific computing, data analysis, time-series analysis and high performance computing. For testing and validating the proposed model and implementation, a case study in astronomy, i.e. the prediction of asteroid's orbital elements, was done. Moreover, a comparison and analysis with the results produced by algorithm of Regularized Mix Variable Symplectic 4 Yarkovsky Effect (RMVS4-YE) is also presented in this paper to provide a high level of confidence on the proposed model.

Keywords: Exponential smoothing, orbital element, parallel computing, R programming language, time series analysis

ARTICLE INFO

Article history:
Received: 25 April 2017
Accepted: 28 November 2017

E-mail addresses:
lala.s.riza@upi.edu, lala_s_riza@yahoo.com (Lala Septem Riza),
j.aria.utama@upi.edu (Judhistira Aria Utama),
syandi.mufti@student.upi.edu (Syandi Mufti Putra),
fmsimatupang@yahoo.com (Ferry Mukharradi Simatupang),
eddypn@upi.edu (Eddy Prasetyo Nugroho)
*Corresponding Author

ISSN: 0128-7680

INTRODUCTION

Currently, the flood of data can no longer be stopped and thus, it inundates every corner of our daily activities. In other words, data have been released in a high amount and at a high speed with complex formats and from various sources. Based on a report from International Data Corporation in 2011, the overall digital

data volume in the world was 1.8 zettabytes, which was expected to grow by nearly nine times within the next five years (Gantz & Reinsel, 2011). A survey by Troester (2015) revealed that Facebook handles more than 250 million photo uploads and the interactions of 800 million active users with more than 900 million objects (pages, groups, etc.) on a daily basis. Wal-Mart processes more than a million customer transactions each hour and imports those into databases that are estimated to contain more than 2.5 petabytes of data. This phenomenon offers two opposing sides. First, it brings emerging problems since available tools and algorithms have difficulties in handling such large data efficiently. At the same time, however, this condition offers great advantages if we are able to extract knowledge from data, such as in making a better decision, predicting a future action, and describing a current situation, etc.

The term 'Big Data' has been widely used for expressing the above phenomenon. It was introduced by computer scientists several years ago. In 2012, Gartner, Beyer & Laney (2012), stated that "Big Data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making." According to this definition, there are 3Vs (which are volume, velocity, and variety) that should be taken into account. Basically, there are two issues related to the volume of Big Data. First, current computers and algorithms cannot handle massive datasets efficiently. Second, current storage management systems are also facing the same issue. In relation to the second issue, which is the aim of this research, there are many techniques that can be used, including data sampling and memory managements (i.e., to create, store, access and manipulate massive matrices) by allocating shared memory and using memory-mapped files (Kane, Emerson, & Weston, 2013), parallel and distributed computing (Zomaya, 1996), and Big Data platform (Dean & Ghemawat, 2008; Murthy, Vavilapalli, Eadline, Niemiec, & Markham, 2013).

In this research, we attempted to design a model and implement it on parallel computing to deal with forecasting large-time series datasets in the R programming language. In order to apply for prediction, we modified exponential smoothing so that it could be run in parallel. In short, we constructed three phases in parallel exponential smoothing, as follows: (i) data pre-processing by utilising the bootstrap method, (ii) conducting exponential smoothing in a parallel way by using three software libraries in R: "forecast", "foreach", and "doParallel", and then (iii) aggregating results to obtain the final predicted values.

Moreover, some experiments are presented to validate the model. These experiments used datasets in astronomy, which are asteroid's orbital elements. The data involved more than 400,000 rows and 6 important parameters in Kepler Components (i.e., a, e, i, ω, Ω and M representing semi major axis, eccentricity, inclination, argument of perihelion, longitude of ascending node and mean anomaly, respectively). The position and velocity of celestial bodies in their orbit at a certain time is expressed by 4 elements orbits. The size and shape of the orbit are represented by the element of a and e, respectively. Meanwhile, the element of $\varpi(=\omega+\Omega)$, which is also known as the longitude of perihelion, specifies the orientation, and the other element, M, specifies the position or phase of celestial bodies in their orbit.

TIME SERIES ANALYSIS

Introduction to Time Series Analysis

Time series datasets can be generated in econometrics, finance, weather station, earthquake, astronomy, medical clinic, energy, and other domains. For example, in finance, we obtained time series datasets produced by the S&P500 daily stock index, as illustrated in Figure 1 below.

Figure 1. S&P500 Daily Stock Index (January 4, 2006, to December 16, 2016)

To study these datasets thoroughly, it is a common way to consider the logarithm return, which is defined as follows (Palma, 2016).

$$r_t = \log \frac{P_t}{P_{t-1}} = \log P_t - \log P_{t-1} \tag{1}$$

where P_t represents the price or the index value at time, t. The log returns are displayed in Figure 2, in which the great volatility could be seen as happening sometime in January 2009.

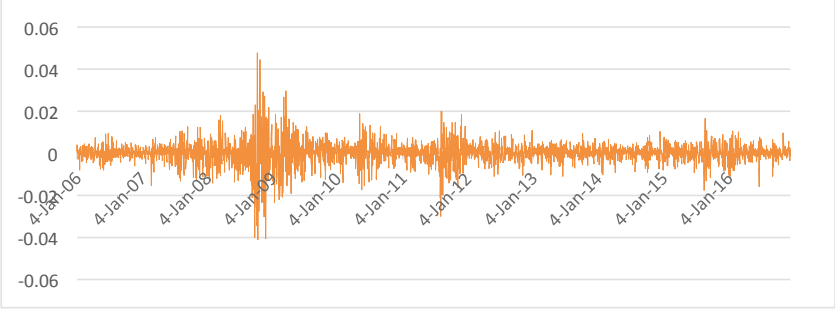

Figure 2. S&P500 Daily Log Returns (4 January 2006 to 16 December 2016)

According to the example above, we can define time series datasets as a time-ordered sequence of observation values of a defined variable at a certain time interval, Δt (Palit & Popovic, 2006). The data can be represented as a set of discrete values (i.e., $x_1, x_2, ..., x_n$). The main properties of time series are stationarity, linearity, trend, and seasonality. The first term means that the mean value and variance of data should be constant over time and the covariance value between x_t and x_{t-d} is dependent on the distance between those two data points and constant over time. Linearity refers to the sequence of observation values can be represented by a linear function. The trend component of time series datasets is a change on the local and global increases or decreases of data values in long-term periods. The last property of time series, which is seasonality, refers to changing on pattern periodically (Palit & Popovic, 2006).

Moreover, time series analysis focused on studying patterns or structures of the datasets so that we could make a better decision and solve the issues on the hand. Basically, the analysis involves the following activities: (i) definition and description of time series, (ii) model construction, (iii) forecasting or prediction of future data, and (iv) clustering to determine the characteristics of data (Riza, 2016b). In this research, we focused on forecasting or predicting of future data by considering historical data. Forecasting can be defined as a given set of observed values $x_1, x_2, x_3, ..., x_n$ of a time series, the future value $x_{n+1}, x_{n+2}, \ldots$, should be estimated (Palit & Popovic, 2006). Moreover, Palit and Popovic (2006) divided strategies on forecasting into the following groups: using trend analysis, regression approaches, the Box-Jenkins methods and smoothing methods. Firstly, the trend analysis utilises linear or nonlinear regression (e.g., quadratic and exponential functions). In regression analysis, we can perform a mathematical tool mapping input variables and its output. Recently, in addition to using approaches on the mathematics field, machine-learning techniques are conducted. For example, a research by Zhang (2012) provides a survey on neural network applications in time series forecasting such as air pollutant concentration, stock index option price, etc. A family of forecasting methods proposed by Box and Jenkins (Box, Jenkins, Reinsel, & Ljung, 2015) consists of Autoregressive Model (AR), Moving-average Model (MA), etc. The last group being focused in this paper, which is smoothing method, is a set of techniques based on reduction of irregularities or random fluctuations in time series data so that we could obtain a clean time series pattern out of contaminated observation data (Palit & Popovic, 2006). Furthermore, useful explanations on time series analysis can be found from in the works of Kirchgässner, Wolters, and Hassler (2012) and Derryberry (2014).

Smoothing Methods for Time Series Forecasting

As mentioned previously, smoothing can be defined as a technique by conducting reduction of irregularities or random fluctuations in time series datasets to obtain a clean time series pattern out of contaminated observation data (Palit & Popovic, 2006). There are two types of smoothing methods, namely moving-average smoothing and exponential smoothing.

The following is a formula of moving-average smoothing for prediction of future values:

$$x_m(t+1) = \frac{x(t) + x(t-1) + \cdots + x(t-n)}{n}. \quad (2)$$

It can be seen that the equation basically averages the past values for reducing the random variations present in *n* observation data. Another variant of the equation is by using weigh on each the past values, as illustrated in the following equation:

$$x_m(t+1) = w_1 x(t) + w_2 x(t-1) + \cdots + w_n x(t-n). \quad (3)$$

Thus, the method can be regarded as easy to understand and simple to use. However, in the case of the weighted moving average, it is difficult to choose the optimal values for each weight of the past values.

The second method, which is exponential smoothing, is a set of techniques where the weights are decreased exponentially as the observations get older (Hyndman, Koehler, Ord, & Snyder, 2008). According to the taxonomy of proposed by Pegels (1969), classification of exponential smoothing methods was obtained. It was improved by Gardner Jr. (1985), modified by Hyndman, Koehler, Snyder, and Grose (2002), and later extended by Taylor (2003), giving the fifteen methods in Table 1 below.

Table 1
Classification of exponential smoothing methods

Trend Component	Seasonal Components		
	N (None)	A (Adaptive)	M (Multiplicative)
N (None)	N, N	N, A	N, M
A (Adaptive)	A, N	A, A	A, M
A_d (Adaptive Damped)	A_d, N	A_d, A	A_d, M
M (Multiplicative)	M, N	M, A	M, M
M_d (Multiplicative damped)	M_d, N	M_d, A	M_d, M

There is another name for some cells such as follows:

- cell (N,N) is the simple exponential smoothing (or SES) method, which is defined as (Brown, 1959):

$$\hat{x}_{t+1} = \hat{x}_t + \alpha(x_t - \hat{x}_t), \quad (4)$$

  where α is a constant between 0 and 1.

- cell (A,N) is named Holt's linear method, which is defined as (Holt, 1957):

  ◦ Level:

$$l_t = \alpha x_t + (1-\alpha)(l_{t-1} + b_{t-1}), \quad (5)$$

- Growth:

$$b_t = \beta^*(l_t - l_{t-1}) + (1 - \beta^*)b_{t-1}, \quad (6)$$

- Forecast:

$$\hat{x}_{t+h|t} = l_t + b_t h, \quad (7)$$

where α and β^* are a contant between 0 and 1.

For a complete list of the exponential smoothing included in Table 1, please refer to Hyndman et al. (2008).

A Brief Survey on the Implementations of Parallel Computing in Time Series Analysis

In this section, we present a brief review on some strategies and implementations of parallelisation of methods in time series analysis, especially for forecasting. It is presented to provide some related works instead of providing a comprehensive survey.

Table 2 shows a summary of the survey conducted in Scopus related to the following keywords: "time series and parallel computing" and "exponential smoothing and large data." It can be seen that researchers have been attempting to find suitable strategies to deal with large datasets or Big Data. For example, Big Data platform, MapReduce, is used in Mirko and Kantelhardt (2013), Sheng, Zhao, Leung and Wang (2013), etc., whereas Message Passing Interface (MPI) is utilised in the research conducted by Górriz, Algeciras, Puntonet, Salmerón, and Martin-Clemente (2004).

INTRODUCTION TO R AND ITS ECOSYSTEM

In this section, we briefly provide an introduction to R and its ecosystem. In addition, some examples of implementations R packages used for parallel computing and time series analysis will also be given in this section. Therefore, some reasons why we are using R in this research are presented.

Introduction to R Programming Language

R is an open-source programming language and software environment used for scientific computing, data analysis, visualisation, time series analysis, high performance computing, etc. (Ihaka & Gentleman, 1996). Furthermore, regarding a survey conducted by KDnuggets (Piatetsky, 2016), R takes the first place for the programming language used for an analytics/ data mining/data science in 2016 and 2015, as shown in Figure 3.

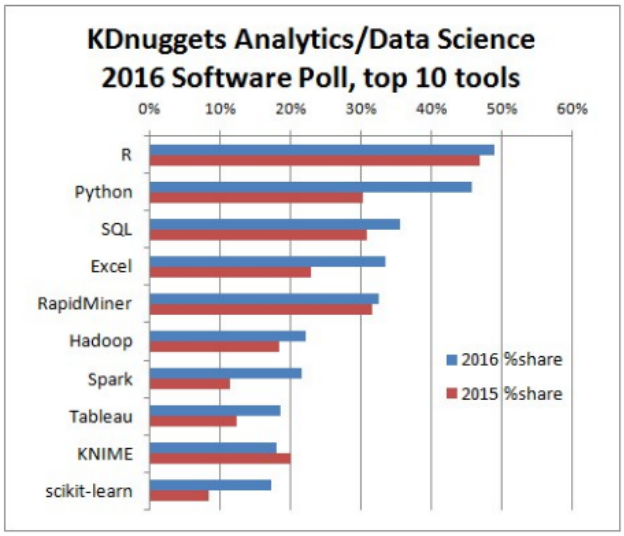

Figure 3. KDnuggets analytics/data science 2016 software poll: top 10 most popular tools (Piatetsky, 2016)

Table 2
A short review on implementations time series analysis in parallel computing

No	Refs	Objectives	Implemented Methods	Strategies for Parallelization
1	Mirko & Kantelhardt, (2013)	To implement statistical time series analysis, such as correlation, autocorrelation, in Big Data/large time series datasets	Correlation and autocorrelation	MapReduce and Hadoop platform
2	Sheng et al. (2013)	To design and implement Echo state network (ESN) for prediction time series by utilizing Extended Kalman Filter (EKF) in parallel computing by using MapReduce	Extended Kalman Filter	MapReduce
3	Górriz et al. (2004).	To implement a Parallel Neural Network (Cross-over Prediction Model) for time series forecasting	Neural network	PVM ("Parallel Virtual Machine") and MPI ("Message Passing Interface"
4	Zhao, Bryan, King, Song, & Yu (2012)	To implement an array-based algorithm to calculate summary statistics for long time-series daily grid climate data sets	Spatial analysis	The Parallel Python (PP) package
5	Liu & He (2012)	To find the optimal segmentation scheme of time series with a low execution time.	a modified Ant Colony Optimization algorithm (WACOS)	OPenMP library in C++

Table 2 *(continue)*

6	Xie, Wulamu, Wang, (2014)	To perform clustering analysis for time series data	Canopy and K-means based on SVD	MapReduce and Hadoop platform
7	Qian et al. (2014)	To present our parallel design of time series analysis and implementation of ARIMA modelling in MADlib's framework.	ARIMA	MADlib on Greenplum database and PostgreSQL database

From the programming language perspective, the R language has some characteristics that render it to offer some advantages (Wickham, 2014) such as providing complete data structures to ease data processing like list, *matrix, vector*, and *data.frame*.

Mostly, packages developed in the R framework are included in the following repositories: CRAN and the Bioconductor project. CRAN (Wickham, 2014) can be found at http://cran.r-project.org/, which is maintained through the efforts of volunteers (the "CRAN team") and the resources of the R Foundation and the employers of those volunteers (WU Wien, TU Dortmund, U Oxford, AT\&T Research). Meanwhile, Bioconductor (http://www.bioconductor.org/) is an open source, open development software project to provide R tools for the analysis and comprehension of high-throughput genomic data. Now, there are over 8000 packages available in CRAN, and these are classified into more than 30 task views. For instance, the task view of Time Series Analysis contains more than 30 R packages such as, the "forecast" package which involves methods and tools for displaying and analysing univariate time series forecasts (Hyndman & Khandakar, 2008).

Parallel Computing in R: "foreach" and "doParallel"

Firstly, we need to define what the definition of parallel computing is. It refers to a kind of computation in which many calculations or the execution of processes is conducted simultaneously (Gottlieb & Almasi, 1989). In other words, the processes need to be divided into some smaller modules so that these parts are executed at the same time. There are at least four types of parallel computing: bit-level parallelism, instruction-level parallelism, task parallelism, and data parallelism.

In the R ecosystem, there are more than 30 packages available at CRAN (https://CRAN.R-project.org/view=HighPerformanceComputing), which are used for high performance and parallel computing with R. In this CRAN Task View, we can find several groups of parallel computing such as Explicit Parallelism (e.g., the "pbdMPI" and "foreach" package), Grid Computing (e.g., the "multiR" package), etc.

The "foreach" package was developed by Calaway et al. (2015). It provides a new looping mechanism for executing R code simultaneously. Hence, the main reason for using the "foreach" package is that parallel computing on multiple processors/cores and multiple nodes of a cluster are available. Moreover, the "doParallel" package has a role as a backend engine for the "foreach" package. It provides a mechanism needed to execute the *foreach* command in parallel. It means that the "foreach" package must be used together with "doParallel" to execute code in parallel.

The following R code is a simple example to show how the "doParallel" and "foreach" perform a task in parallel:

```
R> library(foreach)
R> library(doParallel)
R> registerDoParallel(cores=2)
R> foreach(i=1:3) %dopar% sqrt(i)
```

The code on the first and second lines means we load the following packages: "foreach" and "doParallel". The second line shows that we use multicore-like functionality by defining 2 cores, whereas the last one is that we perform the square root command (i.e., *sqrt()*) of the three objects in parallel with 2 cores. Therefore, it yields the following results, as follows: 1, 1.414214, and 1.732051.

Time Series Analysis in R: "forecast"

The next package used in this research is "forecast." It provides automatic forecasting using exponential smoothing, ARIMA models, and other common forecasting methods (Hyndman & Khandakar, 2008). Related to this research, the following is the signature of one function included in the "forecast" package:

```
holt(y, h=10, damped=FALSE, level=c(80,95), fan=FALSE, initial=c("optimal","simple"),
exponential=FALSE, alpha=NULL, beta=NULL, lambda=NULL, biasadj=FALSE, x=y,
...)
```

The *holt()* function is used for predicting time series datasets based on the exponential smoothing methods. It can be seen that this function has several arguments such as: y is a numeric vector or time series, *h* is number of periods for forecasting/lead time, etc.

Design and Implementation of Parallel Exponential Smoothing

Figure 4 shows the model of parallel exponential smoothing utilising the package "forecast", "foreach", and "doParallel" for forecasting time series data.

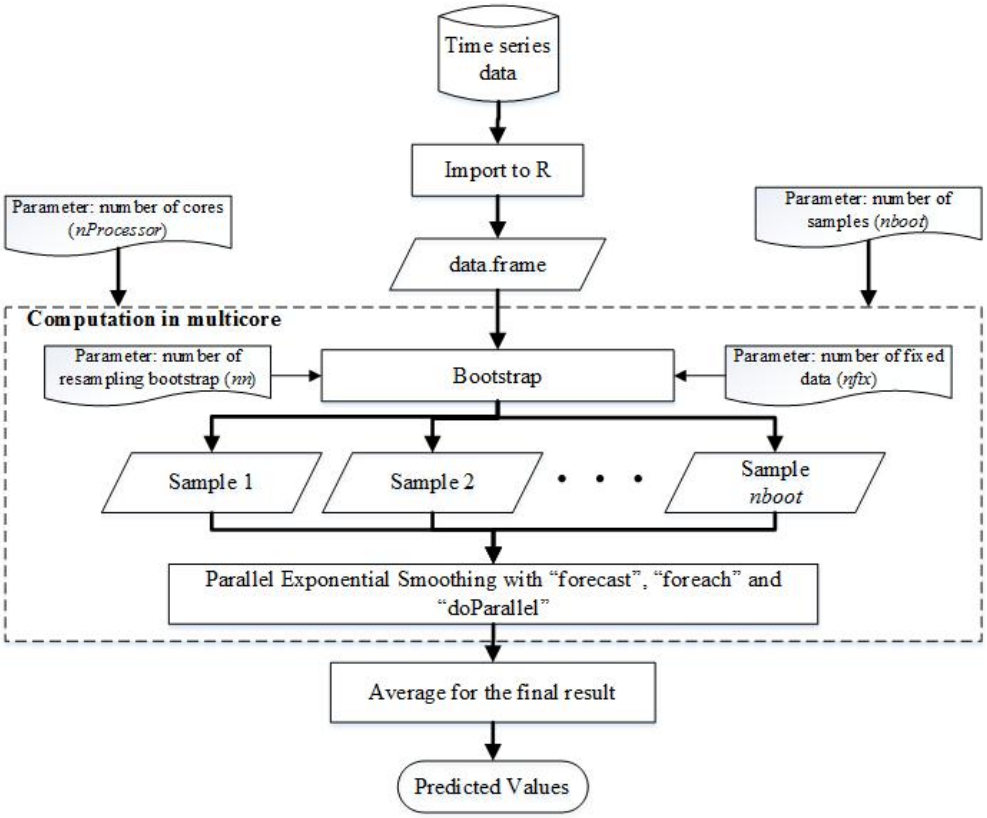

Figure 4. The model of parallel exponential smoothing for forecasting time series data in multicore

It can be seen that basically there are three main steps in this model, as follows:

1. Importing the original data (e.g., *.csv, .xls*, etc.) into an R object, which is *data.frame*. The implementation of this step is as given in the following code:

   R> alldata <- read.table("D:/bootstrap/follow1.out",header=TRUE)

   The *read.table()* function is used for reading the data saved in "D:/bootstrap/follow1.out" and saving into the variable *alldata* as *data.frame*.

2. This step is actually a main part of the model, which is, the computation of bootstrap and exponential smoothing in parallel by utilising the packages: "forecast", "foreach", and "doParallel". In this step, we can divide them into three phases, as follows:
   a. Loading and setting the computation in multicore. Firstly, we load the libraries by executing the following code:

      library(foreach)

      library(doParallel)

After importing the packages, the next code is to define the number of processors/ cores, as follows:

```
R> cl <- makeCluster(nProcessor=2)
R> registerDoParallel(cl)
```

It can be seen from the above code that the number of cores is 2.

b. Developing the procedure of bootstrap on time series data. Bootstrap is a process for selecting some data randomly with replacement so that this sample can represent the whole dataset. It is aimed to make parallelisation by splitting the data. Before explaining the implementation, we depict the processes on the bootstrap as in Figure 5. First, we need to define the number of fixed values (*nfix*) to be the beginning and ending parts of the sample. Then, bootstrap is done to assign the values between the fixed values. The processes will be repeated according to the defined number of sample (*nboot*).

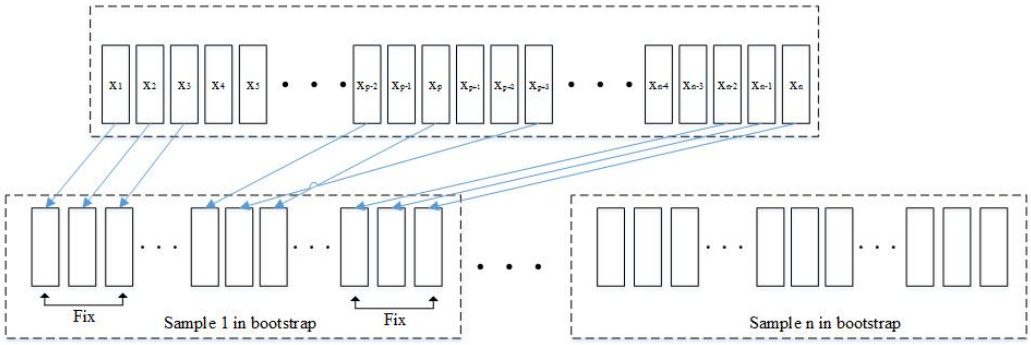

Figure 5. The procedure on the bootstrap method

The implementation of bootstrap can be seen in the following code:

```
boots <- function(alldata,nfix,nn){
    n<-nrow(alldata)
    a<-c(1:nfix)
    z<-c((n-nfix+1):n)
    i<-c(a,sample(c((nfix+1):(n-nfix-1)),nn,replace=T),z)
    x<-c(i[!duplicated(i)])
    ii<-c(1:length(x))
    temp<-list()
    tempData<-NA
    tempData[ii]<-NA
    temp$a<-tempData
```

```
temp$e<-tempData
temp$i<-tempData
temp$O<-tempData
temp$w_omega<-tempData
temp$a[x]<-alldata$a[x]
temp$e[x]<-alldata$e[x]
temp$i[x]<-alldata$i[x]
temp$O[x]<-alldata$O[x]
temp$w_omega[x]<-alldata$w_omega[x]
temp$a<-temp$a[!is.na(temp$a)]
temp$e<-temp$e[!is.na(temp$e)]
temp$i<-temp$i[!is.na(temp$i)]
temp$O<-temp$O[!is.na(temp$O)]
temp$w_omega<-temp$w_omega[!is.na(temp$w_omega)]
return (temp)
}
```

It can be seen that the computation of bootstrap is actually on the fifth line. Then, the process is repeated along with all columns of the data. In this case, we consider five columns, *a, e, i, O*, and *w_omega*, which represent semi major axis, eccentricity, inclination, argument of perihelion, and longitude of ascending node.

c. Prediction using exponential smoothing in parallel. This step is aimed to execute the "forecast" package for forecasting by using exponential smoothing in the "foreach" and "doParallel" package. It can be done by the following code:

```
predValue <- foreach(icount(nboot), .combine=rbind, .export=ls(envir=globalenv()))
%dopar%{
databaru<-boots(alldata,nfix,nn)
 numPoint<-nrow(databaru)
a<-holt(databaru$a,h=nForecast)
e<-holt(databaru$e,h=nForecast)
i<-holt(databaru$i,h=nForecast)
O<-holt(databaru$O,h=nForecast)
w_omega<-holt(databaru$w_omega,h=nForecast)
 return(c(a,e,i,O,w_omega,numPoint,.combine=rbind))
}
```

Thus, after executing the *boots()* function, we call the *holt()* function included in the "forecast" package on each column (i.e., *a, e, i, O,* and *w_omega*) on the *foreach()* function.

3. Average the results for obtaining predicted values. As we have some results that are according to the number of bootstrap samples, it is necessary to calculate the average value. This can be done by executing the *mean()* function built in R.

EXPERIMENTATION ON THE CALCULATION OF ASTEROID'S ORBITAL ELEMENT

In this section, we discuss three topics related to the experiment in this research. Firstly, we briefly explain the problem statement, and then illustrate how we collect the data. Finally, the experimental design is presented.

Problem Statement

Orbit calculations of celestial bodies in astronomy for a simple case is in the form of the two isolated objects (two-body problem), i.e., a single object with a smaller mass is orbiting other object that has a greater mass and under the influence of their mutual gravitational attraction only. Indeed, the two objects are orbiting their common centre of mass, where the orbital velocity and distance of each object from the common centre of mass are determined by each object's mass and their centre-to-centre distance. In the case of more than 2 objects are considered (generally known as the N-body problem), the same equations of motion can be expanded to the number of objects being simulated. In the implementation, the N-body problem is solved using restricted tree-body problem approach, where the third object is considered to have a negligible mass relative to the first and second objects. This approach is quite accurate such as when the systems considered are the Sun-planet-natural/artificial satellite and sun-planet-asteroid/comet as in this work.

The equation of motion for the N-body problem is (see, for example, Murray & Dermott, 1999) as follows:

$$\frac{d^2\boldsymbol{x}_i}{dt^2} = -\sum_{i,j=1;j\neq i}^{N} G\frac{m_j(\boldsymbol{x}_i-\boldsymbol{x}_j)}{|\boldsymbol{x}_i-\boldsymbol{x}_j|^3} \tag{8}$$

In equation (8), G is the universal gravitational constant and on the right hand side is the total gravitational attraction of the whole bodies considered. Equation (8) cannot be solved analytically but numerically for a certain time step in order to obtain a new position vector until the desired time. Some integration techniques available can be employed to obtain the numerical solution of equation (8), such as Wisdom-Holman Mapping (Wisdom & Holman, 1991), Regularised Mix Variable symplectic (Levison & Duncan, 1994), A fourth order T+U symplectic (TU4) method (Candy & Rozmus, 1991), Bulirsch-Stoer method (Press, Teukolsky, Vetterling, & Flannery, 1992), and so on.

Between the orbits of Mars and Jupiter (2.0 to 3.3 Astronomical Unit; 1 AU is defined as the average distance between the Earth-Sun), there is a population of asteroids, space rocks of various sizes that are remnants of the Solar system formation which failed to become a planet due to gravitational perturbations of Jupiter. This area is known as the Main Belt. It is strongly believed that this asteroid population is a major source of near-Earth asteroids (NEAs) (Bottke et al., 2002; Greenstreet, Ngo, & Gladman, 2012). The population of asteroids with orbit such

that [q (asteroids' perihelion distance – the closest distance of asteroids from the Sun) < 1.3 AU and Q (asteroids' aphelion distance – the furthest distance of asteroids from the Sun) > 0.98 AU] so the orbit bring them to near-Earth space. The mechanism that contributes to deliver the asteroids in the Main Belt toward the near-Earth space could be a collision among the asteroids which with the right post-collision velocity and trajectory immediately puts asteroids in the resonance zone (Farinella, Gonczi, Froeschlé, & Froeschlé, 1993), which will further change the orbit to become more elliptical or via slowly drift under the influence of the non-isotropic thermal force (Yarkovsky effect) (Bottke et al., 2002).

While in the near-Earth space, asteroids could experience close encounter with particular planets (Mercury, Venus, Earth and Mars). As a result of this close encounter, the asteroids may be fragmented due to strong tidal force, or if they can survive, their orbit can change drastically in a short time. Drastic changes in orbit occurred can change the fate of the asteroids in the future, i.e. whether they will still be orbiting the Sun or even collide with massive objects in the Solar system. Therefore, observational survey to find the presence of new asteroids and continually observations post-close encounter of asteroids with massive objects is essential to assess the probability of collision with the planets and the Sun in the future. The occurrence of an asteroid's close encounters with massive objects in the Solar System in the future can be predicted by computing the orbit for a certain span of time. Given that the orbits of asteroids in near-Earth space are very chaotic due to their close encounters with planets, orbit computation for asteroids' final fate prediction purpose is commonly done by generating some virtual asteroids (VA) which has a slightly different orbital elements from the real one. Evolution of the real and virtual asteroids' orbital elements obtained during the considered span of time will determine the orbital elements distribution, and where the most heavily populated cells are likely located. These are shown in Figure 6 for asteroid 2012 DA14 (see, for example, Wlodarczyk, 2012; Utama, Dermawan, Hidayat, & Fauzi, 2015) during our 106 years of orbital computation forward experiment after its close encounter with the Earth on 15th February 2013.

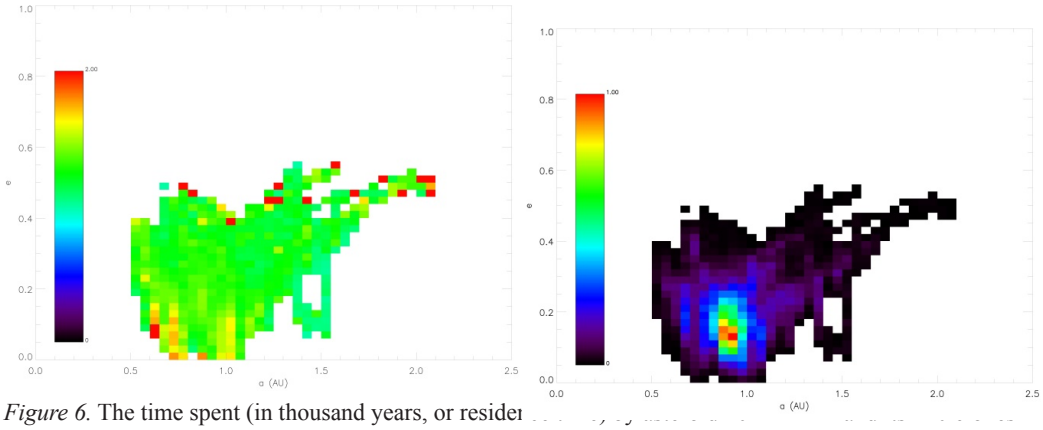

Figure 6. The time spent (in thousand years, or reside[illegible] cell (a, e) during 10^6 years orbital computation represents the expected orbital distribution. The colour scale gives the average time (thousands of years per particle) spent in the different cells; black being the shortest, while red is the longest. White corresponds to unvisited regions (left panel). Number of asteroids fall into each cell (a, e). Red depicts the most frequent cells visited by asteroids, while black depicts cell most rarely visited. White corresponds to unvisited region (right panel)

From the orbital computation, there are 9 clones of 120 that will end their life when colliding with Venus (4 clones) and Earth (5 clones).

Data Gathering

Asteroid's orbital elements data used in this work were obtained from NASA database (http://ssd.jpl.nasa.gov/sbdb.cgi) with the initial value of orbital elements (see Table 3), while description of each orbital element shown in Figure 7. The orbital evolution executed using SWIFT integrator package (Levison & Duncan, 1994) had been modified, namely SWIFT RMVS4YE, which has included non-isotropic thermal force in it (Dermawan, Hidayat, & Utama, 2013). The RMVS4 scheme implemented in this integrator helps us to do computation of orbital elements accurately, especially when a close encounter between asteroid with massive objects occurs. By including the thermal force (Yarkovsky effect), the accurate orbit predictions for long duration ($> 10^5$ years) computation can be reached.

Table 3
Orbital element of Asteroid used as a test case for parallel computing in this work

Orbital Element	Value
a	1.2456 AU
e	0.33552
I	13°.3358
Ω	337°.2327
ω	276°.8451
M	191°.0565

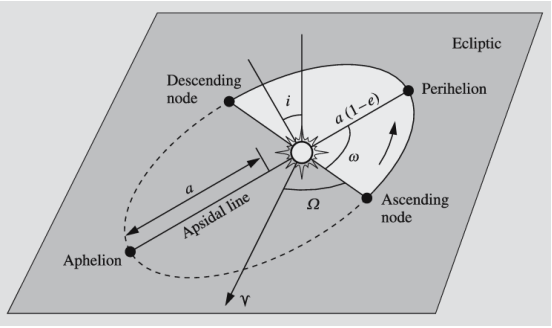

Figure 7. Six integration constants are needed to describe celestial objects' orbit. The constants are the longitude of ascending node Ω, the argument of perihelion ω, the inclination i, the semi major axis a, the eccentricity e and the time of perihelion passage τ [= (M x P/2π) + τ_0, where P is orbital period]
(*Source:* Karttunen, Kröger, Oja, Poutanen, & Donner, 2007)

Experimental Design

In order to do the experiments efficiently, we designed the following two scenarios of prediction: single computing and parallel computing. For the first scenario, we predicted 10 next point of

time, whereas in the second, several simulations were conducted with different as illustrated in Table 4.

Table 4
Parameters on the parallel-computing scenario

Parameters	Description	Values
nProcessor	Number of processors/cores	1 and 4
nboot	Number of samples on bootstrap	4, 12, and 20
Nn	Number of data point on each sample	1000, 10000, and 25000
nForecast	Lead time	1 and 10
nfix	Number of fixed values for the beginning and ending parts of time series data after bootstrap	2 and 20

Moreover, we compared the results with the algorithm RMVS4-YE by calculating Mean Absolute Percentage Error (MAPE) defined, as follows:

$$\text{MAPE} = \frac{\sum \frac{|x_i - f_i|}{x_i}}{n} \times 100\% \quad (9)$$

where x and f are the true and predicted values, while i and n are lead time and number of observations/samples.

RESULTS AND DISCUSSION

As explained previously, in this experiment, there are two scenarios: prediction by using exponential smoothing on single processor and prediction by using parallel exponential smoothing on multicore. Therefore, results of the MAPE calculation in comparison with those obtained from the Algorithm RMVS4-YE are presented in Table 5. In this case, we predicted for 10 periods for five components of asteroid's orbital element: *a, e, i, O, w_omega*,

Table 5
MAPE calculation between exponential smoothing with single processor and the algorithm RMVS4-YE

Period	*A*	*e*	*I*	*O*	*w_omega*	Average (%)
1	0.000809240	0.044868466	0.010442664	0.039505848	0.011069138	0.021339071
2	0.022884474	0.058763748	0.003809124	0.037560265	0.009701406	0.026543803
3	0.037606712	0.104317766	0.063138274	0.070033749	0.008846505	0.056788601
4	0.045043056	0.068458498	0.065173303	0.062258119	0.007350644	0.049656724
5	0.037680306	0.064277164	0.060649023	0.054457305	0.007521384	0.044917036
6	0.037680306	0.055576884	0.058092212	0.063763096	0.013759502	0.045774400
7	0.022958057	0.083037792	0.062095493	0.078176277	0.019996522	0.053252828
8	0.022958057	0.078855233	0.070624528	0.074059252	0.016621183	0.052623651
9	0.023031641	0.076933502	0.073315612	0.062859932	0.008204252	0.048868988
10	0.030392229	0.043376160	0.064199768	0.063112270	0.013159997	0.042848085

representing semi major axis, eccentricity, inclination, argument of perihelion, and longitude of ascending node. The last column is average of MAPE of all parameters. Moreover, the average computation time of all simulation, which is 84.48 seconds, was also calculated.

Then, for the second scenario (i.e., using parallel exponential smoothing) 72 simulations were performed based on the combinations of parameter given in Table 4. For example, Table 6 shows MAPE calculation on the first simulation that assigns the following parameters: *nProcessor* = 1, *nboot* = 4, *nfix* = 2, *nn* = 1000, and *nforecast* = 1. It can be seen that the average of MAPE in this case is around 0.312%.

Table 6
MAPE calculation of the first simulation in the second scenario

a	*E*	*i*	*O*	*w_omega*	Average (%)
0.008680939	0.251733092	0.597202154	0.694654647	0.005684152	0.311590997

Because of the limited space in this paper, we recapitulated the complete results, which are average of MAPE from 72 simulations on the second scenario, as illustrated in Table 7 below.

Table 7
The complete recapitulation of MAPE calculation of 72 simulations on scenario 2

No	*nProcesor*	*nboot*	*nfix*	*nn*	*nforecast*	Average *numPoint*	Computation Cost (s)	MAPE Average (%)
1	1	4	2	1000	1	1003.750	1.7453090	0.31159100
2	1	4	2	1000	10	1003.000	1.8960360	1.94565918
3	1	4	20	1000	1	1039.750	1.7257320	0.15078242
4	1	4	20	1000	10	1038.500	1.6875560	1.05304777
5	1	4	2	10000	1	9908.750	4.5730320	0.04971548
6	1	4	2	10000	10	9902.000	4.8302820	0.28135093
7	1	4	20	10000	1	9939.250	4.4239750	0.02676445
8	1	4	20	10000	10	9940.750	4.4637740	0.14651301
9	1	4	2	25000	1	24394.750	10.5053300	0.02695131
10	1	12	2	25000	10	24403.750	9.9289680	0.12089122
11	1	12	20	25000	1	24441.500	10.1047500	0.02123174
12	1	12	20	25000	10	24426.250	9.1230160	0.09271730
13	1	12	2	1000	1	1002.667	3.1559090	0.41470090
14	1	12	2	1000	10	1002.917	2.8936640	2.07270682
15	1	12	20	1000	1	1039.000	2.9690490	0.15143420
16	1	12	20	1000	10	1039.333	2.9958390	0.82286152
17	1	12	2	10000	1	9907.083	12.6789100	0.04737498
18	1	12	2	10000	10	9903.500	12.9946800	0.27593724
19	1	12	20	10000	1	9941.500	11.6467000	0.02591236
20	1	12	20	10000	10	9943.333	12.6069800	0.15076421
21	1	12	2	25000	1	24378.080	29.9228700	0.02606734

Table 7 *(continue)*

22	1	12	2	25000	10	24394.000	28.3534500	0.12098353
23	1	12	20	25000	1	24421.000	28.0681100	0.02086259
24	1	12	20	25000	10	24439.080	31.4522200	0.09423594
25	1	20	2	1000	1	1002.900	4.4074640	0.39646641
26	1	20	2	1000	10	1003.000	4.8229670	2.14794240
27	1	20	20	1000	1	1039.050	4.4783750	0.16141778
28	1	20	20	1000	10	1039.050	4.3665020	0.85253999
29	1	20	2	10000	1	9902.550	20.7596900	0.04828453
30	1	20	2	10000	10	9903.350	18.8794600	0.26617271
31	1	20	20	10000	1	9943.250	18.5950500	0.02625648
32	1	20	20	10000	10	9937.750	17.6448800	0.14852876
33	1	20	2	25000	1	24388.100	44.4893600	0.02582092
34	1	20	2	25000	10	24388.450	45.3040700	0.12037558
35	1	20	20	25000	1	24423.150	43.3132700	0.02086241
36	1	20	20	25000	10	24429.350	46.4674400	0.09522702
37	4	4	2	1000	1	1003.000	3.2107390	0.45777004
38	4	4	2	1000	10	1002.750	3.0782100	2.22245284
39	4	4	20	1000	1	1039.000	3.0190890	0.15731712
40	4	4	20	1000	10	1038.500	3.1023910	0.82419055
41	4	4	2	10000	1	9899.000	4.8375500	0.04813014
42	4	4	2	10000	10	9892.750	4.7776220	0.26066381
43	4	4	20	10000	1	9934.250	4.6702120	0.02545224
44	4	4	20	10000	10	9942.250	4.5244820	0.14652436
45	4	4	2	25000	1	24399.750	7.5436390	0.02593090
46	4	12	2	25000	10	24403.000	6.8495680	0.11601128
47	4	12	20	25000	1	24420.500	7.1595970	0.02139856
48	4	12	20	25000	10	24450.000	7.4199600	0.09610011
49	4	12	2	1000	1	1002.583	3.6817990	0.40538402
50	4	12	2	1000	10	1003.333	3.6715830	2.21894915
51	4	12	20	1000	1	1039.333	3.7104400	0.16964136
52	4	12	20	1000	10	1038.833	3.6743280	0.82289333
53	4	12	2	10000	1	9903.750	8.2461690	0.04858334
54	4	12	2	10000	10	9907.333	8.5302590	0.27667333
55	4	12	20	10000	1	9940.250	7.5553620	0.02569687
56	4	12	20	10000	10	9937.583	8.1135430	0.27667333
57	4	12	2	25000	1	24386.920	15.5334500	0.02618081
58	4	12	2	25000	10	24390.670	15.8121500	0.12152408
59	4	12	20	25000	1	24429.580	16.7012900	0.02051017
60	4	12	20	25000	10	24424.250	15.8905900	0.09819543
61	4	20	2	1000	1	1002.900	4.4920760	0.37927788
62	4	20	2	1000	10	1001.950	4.4137490	2.17249141
63	4	20	20	1000	1	1039.350	4.3392870	0.15577406

Table 7 *(continue)*

64	4	20	20	1000	10	1039.050	4.4213620	0.88915073
65	4	20	2	10000	1	9904.150	12.3716900	0.04753417
66	4	20	2	10000	10	9903.900	12.0911500	0.29628936
67	4	20	20	10000	1	9937.800	11.0131900	0.02718215
68	4	20	20	10000	10	9942.750	11.2667100	0.15365591
69	4	20	2	25000	1	24388.600	24.6162400	0.02656839
70	4	20	2	25000	10	24391.200	23.8743000	0.12202390
71	4	20	20	25000	1	24423.650	23.8025900	0.02093147
72	4	20	20	25000	10	24432.800	25.6407900	0.09519576

As shown in Table 7, it can be stated that the best results and computation cost is the 72nd simulation, where the MAPE average is about 0.095% at the computation cost around 25.6 seconds. Of course, it is also faster than the first scenario (i.e., exponential smoothing with single processor), where its computation cost is about 84.48 seconds. Moreover, the MAPE average of both simulation is relatively close, which is 0.095% is for the 72nd simulation in the second scenario and 0.044% for the first scenario. This means that the result of parallel exponential smoothing is reasonable. We also focused on analysing the second scenario. For example, the result shows that the higher number of bootstrap causes longer computation time, while higher number of resampling bootstrap makes the lower MAPE, even though certain number of samples cannot make MAPE significantly better.

CONCLUSION AND RECOMMENDATIONS FOR FUTURE WORK

In this research, a model of time series analysis for conducting exponential smoothing with the bootstrap methods in parallel computing by utilising the R packages: "forecast," "foreach," and "doParallel" has been presented. The next contribution is that the implementation of the proposed model has been technically explained in detail so that the code is reproducible. Furthermore, we conducted some experiments for forecasting asteroid's orbital elements in order to evaluate the model and its implementation, along with the analysis of the results.

As for future work, we plan to convert the problem of time series analysis on asteroid's orbital elements into regression one. Then, we will solve it by utilising machine-learning methods, such as fuzzy rule based systems (Riza, Bergmeir, Herrera, & Benitez, 2015), rough set theory and fuzzy rough set theory (Riza et al., 2014; Nazir, Shahzad, & Riza, 2016b), and gradient descent (Riza, Nasrulloh, Junaeti, Zain, & Nandiyanto, 2016a).

REFERENCES

Beyer, M. A., & Laney, D. (2012). *The importance of 'Big Data': A definition.* Retrieved from https://www.gartner.com/doc/2057415/importance-big-data-definition.

Bottke, W. F., Morbidelli, A., Jedicke, R., Petit, J. M., Levison, H. F., Michel, P., & Metcalfe, T. S. (2002). Debiased orbital and absolute magnitude distribution of the near-Earth objects. *Icarus, 156*(2), 399-433.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. New Jersey: John Wiley & Sons.

Brown, R. G. (1959). *Statistical forecasting for inventory control*. New York: McGraw-Hill.

Calaway, R., Analytics, R., & Weston, S. (2015). *Foreach: Provides foreach looping construct for R*. Retrieved from https://cran.r-project.org/web/packages/foreach/index.html.

Candy, J., & Rozmus, W. (1991). A symplectic integration algorithm for separable hamiltonian functions. *Journal of Computational Physics, 92*(1), 230-256.

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM, 51*(1), 107-113.

Dermawan, B., Hidayat, T., & Utama, J. A. (2013). *Pengembangan integrator swift_rmvs4 dengan melibatkan efek termal*. Prosiding Seminar Himpunan Astronomi Indonesia.

Derryberry, D. R. (2014). *Basic data analysis for time series with R*. New Jersey: John Wiley & Sons.

Farinella, P., Gonczi, R., Froeschlé, C., & Froeschlé, C. (1993). The injection of asteroid fragments into resonances. *Icarus, 101*(2), 174-187.

Gantz, J., & Reinsel, D. (2011). *Extracting value from chaos*. Retrieved from http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf.

Gardner, E. S. (1985). Exponential smoothing: the state of the art. *Journal of Forecasting, 4*(1), 1-28.

Górriz, J. M., Algeciras, E. P. S., Puntonet, C. G., Salmerón, M., & Martin-Clemente, R. (2004). Parallelization of time series forecasting model. In *Proceeding of 12th Euromicro Conference* (pp. 103-110). Vienna, Austria: Ustrian Computer Society.

Gottlieb, A., & Almasi, G. S. (1989). *Highly parallel computing*. Redwood City, Calif.: Benjamin/Cummings.

Greenstreet, S., Ngo, H., & Gladman, B. (2012). The orbital distribution of near-Earth objects inside Earth's orbit. *Icarus, 217*(1), 355–366.

Holt, C. C. (1957). *Forecasting trends and seasonals by exponentially weighted averages*. O.N.R. Memorandum 52/1957, Carnegie Institute of Technology.

Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software, 26*(3), 1-22.

Hyndman, R., Koehler, A. B., Ord, J. K., & Snyder, R. D. (2008). *Forecasting with exponential smoothing: the state space approach*. Berlin: Springer Science & Business Media.

Hyndman, R. J., Koehler, A. B., Snyder, R. D., & Grose, S. (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting, 18*(3), 439-454.

Ihaka, R., & Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics, 5*(3), 299-314.

Kane, M. J., Emerson, J., & Weston, S. (2013). Scalable strategies for computing with massive data. *Journal of Statistical Software, 55*(14), 1-19.

Karttunen, H., Kröger, P., Oja, H., Poutanen, M., & Donner, K. J. (2007). *Fundamental Astronomy* (5th Ed). Berlin: Springer.

Kirchgässner, G., Wolters, J., & Hassler, U. (2012). *Introduction to modern time series analysis.* Berlin: Springer Science & Business Media.

Levison, H. F., & Duncan, M. J. (1994). The long-term dynamical behavior of short-period comets. *Icarus, 108*(1), 18–36.

Liu, H., & He, Z. (2012). Parallel ant colony optimization algorithms for time series segmentation on a multi-core processor. In *Proceeding of 2012 4th International Conference of Intelligent Human-Machine Systems and Cybernetics (IHMSC)* (pp. 340-343). Nanchang, China: IEEE.

Mirko, K., & Kantelhardt, J. W. (2013). Hadoop. TS: large-scale time-series processing. *International Journal of Computer Applications, 74*(17), 1-8.

Murray, C. D., & Dermott, S. F. (1999). *Solar system dynamics.* Cambridge: University Press.

Murthy, A. C., Vavilapalli, V. K., Eadline, D., Niemiec, J., & Markham, J. (2013). *Apache Hadoop YARN: Moving beyond MapReduce and batch processing with Apache Hadoop 2.* New Jersey: Pearson Education.

Nazir, S., Shahzad, S., & Riza, L. S. (2016). Birthmark-based software classification using rough sets. *Arabian Journal for Science and Engineering, 42*(2), 859-871.

Palit, A. K., & Popovic, D. (2006). *Computational intelligence in time series forecasting*. London: Springer-Verlag London Limited.

Palma, W. (2016). *Time series analysis.* New Jersey: Wiley.

Pegels, C. C. (1969). Exponential forecasting: Some new variations. *Management Science, 15*(5), 311-315.

Piatetsky, G. (2016). *R, python duel as top analytics, data science software – kdnuggets 2016 software poll results.* Retrieved from http://www.kdnuggets.com/2016/06/r-python-top-analytics-data-mining-data-science-software.html.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in FORTRAN77*. Cambridge: Cambridge University Press.

Qian, H., Yang, S., Iyer, R., Feng, X., Wellons, M., & Welton, C. (2014). Parallel time series modeling-a case study of in-database big data analytics. In *Proceeding of Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 417-428). Berlin: Jerman Springer International Publishing.

Riza, L. S., Bergmeir, C., Herrera, F., & Benitez, J. M. (2015). frbs: fuzzy rule-based systems for classification and regression in R. *Journal of Statistical Software, 65*(1), 1-30.

Riza, L. S., Janusz, A., Bergmeir, C., Cornelis, C., Herrera, F., Slezak, D., & Benıtez, J. M. (2014). Implementing algorithms of rough set theory and fuzzy rough set theory in the R package RoughSets. *Information Sciences, 287*, 68-89.

Riza, L. S., Nasrulloh, I. F., Junaeti, E., Zain, R., & Nandiyanto, A. B. D. (2016a). gradDescentR: An R package implementing gradient descent and its variants for regression tasks. In *Proceeding of 1st International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)* (pp. 125-129). Yogyakarta, Indonesia: IEEE.

Riza, L. S., Wihardi, Y., Nurdin, E. A., Ardi, N. D., Asmoro, C. P., Wijaya, A. F. C., & Nandiyanto, A. B. D. (2016b). Analysis on atmospheric pressure, temperature, and wind speed profiles during total solar eclipse 9 March 2016 using time series clustering. *Journal of Physics: Conference Series, 771*(1), 012009.

Sheng, C., Zhao, J., Leung, H., & Wang, W. (2013). Extended kalman filter based echo state network for time series prediction using mapreduce framework. In *Proceeding of Ninth International Conference of Mobile Ad-hoc and Sensor Networks (MSN)* (pp. 175-180). Dalian, China: IEEE.

Taylor, J. W. (2003). Exponential smoothing with a damped multiplicative trend. *International Journal of Forecasting, 19*(4), 715–725.

Troester, M. (2015). *Big data meets big data analytics.* Retrieved from http://www.sas.com/resources/whitepaper/wp_46345.pdf.

Utama, J. A., Dermawan, B., Hidayat, T., & Fauzi, U. (2015). Dinamika orbit 2012 DA14 pascapapasan dekat dengan bumi. *SPEKTRA: Jurnal Fisika dan Aplikasinya, 16*(1), 1-5.

Wickham, H. (2014). *Advanced R.* Florida: CRC Press.

Wisdom, J., & Holman, M. (1991). Symplectic maps for the n-body problem. *The Astronomical Journal, 102*(4), 1528-1539.

Wlodarczyk, I. (2012). The potentially dangerous asteroid 2012 DA14. *Monthly Notices of the Royal Astronomical Society, 427*(2), 1175-1181.

Xie, Y., Wulamu, A., Wang, Y., & Liu, Z. (2014). Implementation of time series data clustering based on SVD for stock data analysis on hadoop platform. In *Proceeding of 2014 9th IEEE Conference on Industrial Electronics and Applications* (pp. 2007-2010). Hangzhou, China: IEEE.

Zhang, G. P. (2012). Neural networks for time-series forecasting. In *Proceeding of Handbook of Natural Computing* (pp. 461-477). Berlin, Germany: Springer Berlin Heidelberg.

Zhao, G., Bryan, B. A., King, D., Song, X., & Yu, Q. (2012). Parallelization and optimization of spatial analysis for large scale environmental model data assembly. *Computers and Electronics in Agriculture, 89*, 94-99.

Zomaya, A. Y. (1996). *Parallel and distributed computing handbook* (Vol. 204). New York: McGraw-Hill.