METHODS FOR DAILY FORECASTING

by

Özge Tuğrul

B.S., Industrial Engineering, İstanbul Technical University, 2008

Submitted to the Institute for Graduate Studies in

Science and Engineering in partial fulfillment of

the requirements for the degree of

Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2010

Dedicated to my parents

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis advisor, Assoc. Prof. Wolfgang Hörmann for his patience and invaluable guidance. Also, I would like to mention his continuous help, encouragement and constructive criticism during the preparation process of my Master's thesis.

I would especially like to thank Prof. Ali Rıza Kaylan and Assoc. Prof. Şevket Günter for their invaluable contributions and advices on my Master's thesis.

I would like to thank Antalya Provincial Airport Civil Administration for their help.

I would also like to thank TÜBİTAK for their support by providing scholarship during my Master's programme studies.

Finally, I am grateful to my parents for their patience and support.

ABSTRACT

METHODS FOR DAILY FORECASTING

For many branches of business daily forecasting constitutes an important activity, since future plans are made or decisions are taken according to the future expectations. This thesis provides a comparison of different daily forecasting methods with special implementations for different time series data and evaluates the results of forecasting experiments. It is important for the selected forecasting method to give more accurate results and smaller forecast errors. Also, the coding and the implementation processes have an effect in choosing a forecasting method.

The first chapter explains forecasting, the second chapter gives information about the forecasting process, the third chapter gives examples of the forecasting methods used in the literature. The explanation of forecasting methods begins with the fourth chapter with the explanation of the so called “naive method”. The fifth chapter explains time series regression methods, the sixth chapter presents exponential smoothing methods, the seventh chapter gives information about the autoregressive integrated moving average method. The eight chapter explains the implementation of these forecasting methods written on “R” program. The experiments evaluating these forecasting methods are introduced and explained in the ninth chapter. Finally, a general overview of the methods is presented.

ÖZET

GÜNLÜK VERİ TAHMİNLEME YÖNTEMLERİ

Birçok iş kolunda geleceğe yönelik planlar ve alınacak kararlar gelecekteki beklentilere göre şekillendiğinden, tahminleme büyük önem taşımaktadır. Bu tez günlük zaman serisi verileri için özel birtakım uygulamalarla geliştirilmiş tahminleme yöntemlerinin farklı tipte zaman serisi verilerine uygulanarak karşılaştırılmasını ve tahminleme deney sonuçlarının değerlendirmesini gerçekleştirmektedir. Bir tahminleme metodunun gerçeğe yakın tahmin değerleri vermesinin yanında, bu metodun kodlama ve uygulama aşamaları da önem teşkil etmektedir.

Tezin birinci bölümünde tahminleme açıklanmış, ikinci bölümünde, tahminleme sürecinden bahsedilmiştir. Üçüncü bölümde literatürde uygulanmış tahminleme metotlarından bahsedilmiştir. Tezde uygulanacak olan metotların açıklamalarına basit tahminleme yöntemi ile dördüncü bölümde başlanmıştır. Beşinci bölüm zaman serisi regresyon metotlarını, altıncı bölüm ise çeşitli üstel düzeltme yöntemlerini açıklamaktadır. Yedinci bölümde otoregresif hareketli ortalama metodu hakkında açıklamalar ve uygulanan günlük tahminleme metodu yer almaktadır. Sekizinci bölümde, "R" programında yazılmış tahminleme yöntemlerinin uygulanmasına yönelik fonksiyonların açıklamaları yapılmıştır. Tahminleme metotlarını değerlendirmeye yönelik deney sonuçları ve değerlendirmeler dokuzuncu bölümde verilmiştir. En son bölümde ise, yöntemler hakkında genel bir değerlendirme yapılıp, sonuca varılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

X_t	Time series value at time t
λ	Time series transformation parameter
$e_t(n)$	Forecast error of n-period-ahead forecast for time t
$\widehat{Y_t}(t-n)$	Forecast value for time t as of time $(t-n)$
Y_t	Observed value at time t
ME	Mean error
MSE	Mean square error
MAD	Mean absolute deviation
MPE	Mean percentage error
MAPE	Mean absolute percentage error
β_0	The level coefficient in time series regression model
β_1	The slope coefficient in time series regression model
ε_t	Correlated error term for time series regression model
v_t	Independent error term for time series regression model
ρ	Correlation term
α	Level parameter for exponential smoothing
β	Trend parameter for exponential smoothing
γ	Seasonal parameter for seasonal exponential smoothing
ω	Additional seasonal parameter for double seasonal exponential smoothing
φ	Autocorrelation parameter for exponential smoothing
δ	Error term coefficient
AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion
ACF	Autocorrelation Function

ARIMA	Autoregressive integrated moving average
θ	Dampening parameter for exponential smoothing method
ϕ	Autocorrelation parameter for the autoregressive (AR) process of ARIMA
Δ	Error term coefficient for moving average (MA) process of ARIMA
μ	Constant mean of the moving average (MA) process
L_0	Initial level for exponential smoothing
T_0	Initial trend for exponential smoothing

1. INTRODUCTION

Forecasting has been an important issue in fields like economics, trade, production and tourism from the past to the present. There are many time series important in our daily lives. The number of patients coming to a hospital, the number of tourists booking hotel rooms, electricity consumption, cash withdrawals from an automatic teller machine, passenger numbers in a bus station or airport can be given as examples. In many sectors, these events occur generally with a periodicity or seasonal variation. Forecasting future events constitutes an important issue since being aware of future occurrences may provide a better short term and long term allocation of resources. Forecasting thus helps to take operational or strategic decisions and to decrease costs.

Often, yearly, monthly or weekly forecasts are obtained, but in many applications, daily forecasts are required as well. It is not astonishing that monthly or yearly time series reveal less variability since; these types of data are aggregation of daily data. So, the daily forecasting process often needs an extra effort to understand the structure, seasonality and randomness of the time series data. Since many applications require daily forecasts, reliable daily forecasting methods are required to reach a good forecasting performance.

In this thesis, three different daily time series data sets are investigated and different forecasting methods are applied by using the software package “R”. One of the data consists of daily flight counts including both arrivals and departures for different regions; the second data set is daily passenger counts at an airport, and the third data set gives the daily cash withdrawal and deposit amounts from different branches. The data used are real data and are not modified. Forecasts of daily flight numbers are required since employees, maintenance, and repair operations are all scheduled according to forecasted values. On the other hand, the forecasts of cash deposits and withdrawals help a bank to decrease its costs and increase its customer satisfaction by planning its operations for the future periods.

In addition, many daily forecasting methods are coded on “R”. They are based on four main forecasting methods: the naive method, the regression method, the (seasonal) ARIMA method and the exponential smoothing method. These methods are adapted for

daily forecasting with special implementations and also compared with daily forecasting methods used in the literature. The reason for applying many different forecasting methods is to find a simple and precise daily forecasting method for each of the seasonal and daily time series, for each forecast horizon considered.

Chapter 2 explains the general forecasting process. Chapter 3 gives information about the daily forecasting methods used in the literature. Chapter 4 gives information about the naive method and its application to daily time series. Chapter 5 explains the regression method for time series and special regression methods for daily time series. Chapter 6 presents different exponential smoothing methods used in the literature and illustrates adaptations of these methods to daily time series data. Chapter 7 discusses the adaptation of seasonal ARIMA methods to daily time series data. Chapter 8 explains our R functions. Primary findings are provided in Chapter 9 which reports the results on the performance of different forecasting methods for the time series data studied. The thesis ends with the conclusions.

2. THE GENERAL FORECASTING PROCESS

2.1. Forecasting Steps

Forecasting is prediction of future events. Forecasting processes are widely used in many branches of business and industry, government, economics, environmental sciences, medicine, social sciences, politics and finance for short term, medium term and long term prediction of the future. Short term forecasting includes prediction of a few days, weeks or months into the future. In general, medium term forecasting covers prediction of one to two years into the future and long term forecasting extends to prediction of many years into the future. The demand in most sectors is subject to trends, and generally follows weekly, monthly or annual cycles and outliers and interventions. In addition to these complexities, the recorded data type can also change over time, and not only different reporting intervals, but also instantaneous, cumulative or historical data can be used.

Quantitative forecasting techniques make formal use of historical data and a forecasting model. The model formally summarizes patterns of the data and expresses a statistical relationship between previous and current values of the variable. Then the model is used to project the patterns of the data into the future. Some of the better-known quantitative forecasting techniques are regression methods, moving average and smoothing methods, decomposition methods, and the Box-Jenkins (ARIMA) methodology (Montgomery, 2008).

The general forecasting procedures can be described by the following steps:

i. Problem definition

ii. Data collection

iii. Data analysis

iv. Model selection and fitting

v. Model validation

vi. Forecasting and model deployment

vii. Monitoring forecasting model performance

Steps iii-vii are implemented repeatedly, each time Step vii reveals forecast performance which could and should be improved.

In the problem definition part, the frequency of the desired forecasts such as days, hours, weeks, months is decided. Also, the "forecasting horizon" which is the number of future periods for which forecasts should be produced is determined. In addition, the forecast process interval (the frequency with which the forecasting process must be repeated) may be pre-determined. In problem definition step, an appropriate statistical or decision-theoretic metric to measure the accuracy ("goodness") of the forecasts is also defined.

The data collection step consists of obtaining the relevant history for the variables that are to be forecasted. The data analysis step consists of analyzing and inspecting the patterns such as trends, seasonality, and cyclical components. Model selection and fitting consists of choosing one or more forecasting models and fitting the model to the data.

Model validation consists of an evaluation of the forecasting model to determine how it is likely to perform in the intended application. Solely evaluation of the model fit to the historical data is not sufficient. Therefore, the magnitude of the forecast errors should be examined when the model is used to forecast new data. The fitting errors are always smaller than the forecast errors, so data splitting should be applied to the data to obtain fitting segment ("history data") and forecasting segment ("future data"). This can provide useful guidance on how the forecasting model will perform when exposed to new data and can be a valuable approach for discriminating between competing forecasting models and methods.

Forecasting model deployment involves the start of the "routine" use of the model and the resulting forecasts. Monitoring the model performance should be an ongoing activity after the model has been deployed to ensure that it keeps performing satisfactorily. Monitoring reveals forecast performance if it could or should be improved, alternative forecasting methods should be evaluated with their corresponding models re-identified and re-estimated.

2.2. Transformations of Time Series Data

Most of the forecasting methods are linear functions of the available data. However, it should be taken into consideration that there can be also non linear relationships between these data. For certain time series methods, transformations are needed in order to stabilize the variance or to normalize the errors. Bisgaard and Kulahci (2008) suggest that for any statistical modelling, the right transformation leads to more accurate forecasts and better fit of the data. In most of the cases, a log transformation is applied to obtain variance-stationary time series, but in some situations, log transformation may not be appropriate.

Bisgaard and Kulahci (2008) denote that the main purpose of data transformation is to remove the dependence of the variance to the mean of the data, to increase the conformance of the additive models to the data and to obtain normally distributed error terms with zero mean and constant variance. It should be also noted that in order to benefit from data transformation, the ratio of the maximum value of the data to the minimum value of the data should be larger than three.

The Box-Cox transformation is defined by:

$$X_t(\lambda) = \frac{(X_t^{\lambda}-1)}{\lambda} \quad \text{if } \lambda \neq 0. \tag{2.1}$$

$$X_t(\lambda) = logX_t \quad \text{if } \lambda = 0. \tag{2.2}$$

For values of lambda between zero and one, the transformation lies between the logarithmic transformation and identity. One of the ways of determining the proper transformation is to observe the range versus mean chart of the transformed data. If the ranges of the data does not change with different means, then, the transformation can be assumed to be appropriate.

Newbold and Bos (1994) claim that the application of power transformations is not precisely necessary. Some of the forecasting models provide accurate forecast results without any data transformation; however, selecting a good transformation of the data may help to prevent the dilemma of finding the right model. In some situations, when it cannot be decided if there is a multiplicative or additive seasonality, the right transformation may lead to better forecasting results than either the additive or the multiplicative model. So, the right transformation of the data will decrease the effort of choosing the best forecasting model.

2.3. Evaluating Forecasting Model Performance

Generally, it is tempting to evaluate the performance of a model on the basis of the fit of the forecasting or time series model to historical data. The goodness of fit approach often uses the residuals and does not really reflect the capability of the forecasting technique to successfully predict future observations. Montgomery et al. (2008) claim that the user of the forecast is very concerned about the accuracy of future forecasts, not the model goodness of fit, so it is important to evaluate this aspect of any recommended technique. Sometimes, the forecast accuracy is called “out-of-sample” forecast error, to distinguish it from the residuals that arise from a model fitting process. Concentrating too much on the model that produces the best historical fit often results in over-fitting, or including too many parameters or terms in the model just because these additional terms improve the model fit. In general, if we are interested in obtaining one-step-ahead forecasts, the best approach is to select the model that results in the smallest standard deviation of the one-step-ahead forecast errors (or mean squared error) when the model is applied to data that was not used in the fitting process. A standard way to measure this out-of-sample performance is by utilizing some form of data splitting: that is, we divide the time series data into two segments - one for model fitting and the other for performance

testing. When more than one forecasting technique seems reasonable for a particular application, this forecast accuracy measurement procedure can also be used to discriminate between competing techniques.

The lead one forecast error can be defined by:

$$e_t(1) = y_t - \hat{y}_t(t-1) \tag{2.3}$$

where $\hat{y}_t(t-1)$ denotes the forecast of y_t that was made one period prior. Forecast errors at other lags, or at several different lags, could be used if interest focused on those particular forecasts.

Suppose that there are n observations for which forecasts have been made and n one step ahead forecast errors, $e_t(1)$, $t = 1,2,\dots,n$ are obtained. A possible measure of the overall forecast accuracy is the average error or mean error. Mean error is defined by:

$$ME = \frac{1}{n}\sum_{t=1}^{n} e_t(1) \tag{2.4}$$

We hope that the mean forecast error is an estimate of the expected value of forecast error, which we would hope to be zero; that is, the forecasting technique produces unbiased forecasts. If the mean forecast error differs appreciably from zero, the forecasts are biased. If the mean forecast error drifts away from zero when the forecasting technique is in use, this can be an indication that the underlying time series has changed in some fashion, the forecasting technique has not tracked this change, and now biased forecasts are being generated. This is an example of a situation where an alternative forecasting technique or model may need to be identified or the model currently in use may need to be re-estimated.

The mean absolute deviation (or mean absolute error) is defined by:

$$MAD = \frac{1}{n}\sum_{t=1}^{n} |e_t(1)| \tag{2.5}$$

If forecast errors are normally distributed with mean zero, the MAD is related to the standard deviation of the forecast error by:

$$\hat{\sigma}_{e(1)} = \sqrt{\frac{\pi}{2}} MAD \cong 1.25MAD \tag{2.6}$$

The mean squared error is defined by:

$$\hat{\sigma}^2_{e(1)} = MSE = \frac{1}{n}\sum_{t=1}^{n}[e_t(1)]^2 \tag{2.7}$$

Both the mean absolute deviation (MAD) and the mean squared error (MSE) measure the variability in the unbiased forecast errors. Obviously, we want the variability in forecast errors to be small. The MSE is a direct estimator of the one step ahead forecast errors.

The one step ahead forecast error and its summary measures, the ME, MAD, and MSE are all scale dependent measures of forecast accuracy; that is their values are expressed in terms of the original units of measurement. Furthermore, accuracy measures that are scale dependent do not facilitate comparisons of a single forecasting technique across different time series, or comparisons across different time periods. We therefore need a measure of scale-independent forecasting errors.

In general the m step ahead relative forecasting error can be denoted by:

$$re_t(n) = \left(\frac{y_t - \hat{y}_t(t-m)}{y_t}\right) 100 = \left(\frac{e_t(m)}{y_t}\right) 100 \tag{2.8}$$

The mean percent forecast error (MPE) is:

$$MPE = \frac{1}{n}\sum_{t=1}^{n} re_t\,(m) \tag{2.9}$$

and the mean absolute percentage error (MAPE) is:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}|re_t(m)| \tag{2.10}$$

We will use MAPE and MSE for evaluating the forecasting model performance in this thesis. MAPE and MSE for forecast lead times will be obtained. The reason for choosing the mean absolute percentage error is that in this mentioned method, absolute value of the relative errors are taken, so, both negative and positive relative errors are taken into consideration. Also, this measure provides a scale-independent measure of forecast errors. Additionally, we will try to minimize MSE in forecasting methods.

3. LITERATURE REVIEW FOR DAILY FORECASTING

In most of the papers about time series and forecasting methodologies, generally aggregated data such as yearly, monthly and weekly data are analyzed and different forecasting methods are applied to these data. Forecasting the seasonal daily product or service demands of the companies has long been an important issue, as operations management such as scheduling and control of the systems or optimal utilization of the assets and increase in the profits require accurately predicted and detailed information about the future situation of their demands. Daily forecasting provides more detailed information. The reason why there are not many studies on short term or medium term daily forecasting may be related to the characteristics of the seasonal daily data, which require special treatment due to involving weekly, monthly as well as yearly seasonality, plus special day effects such as public holidays, leap years, and unexpected events (outliers or interventions) such as natural disasters. It is more difficult to forecast daily time series than to forecast aggregated time series.

One important issue is to compare different forecasting methods and data transformations in order to determine the best methodology to forecast short term daily demand for different data sets showing different characteristics.

There are a restricted number of articles on practical daily forecasting, and most of them include hourly time series data with only a-few-days-ahead forecasts. In Cranage's (2003) article called "Practical time series forecasting for the hospitality manager", daily seasonality is modelled by three methods. One of the methods was using a 28 days of seasonality periodicity with an assumption that the pattern of sales repeated itself every 28 days within a year. The second method was using a seven day seasonal period. The third approach was using the same 28 days of each year. The difference of this last model from the previous 28-day period model is that this model was based on the same 28 days within each year. In this way, the number of months per year was taken as 13 in order to accommodate the remaining days in a year. The MAPE's calculated using the seven day seasonality were reported to be greater than 25 percent, while using the seasonality based on the same 28 day period each year (with 13 additive and multiplicative seasonality

factors) were noted to have MAPE's between 10 percent and 15 percent. In Cancelo, Espasa and Grafe's (2008) article called "Forecasting the electricity load from one day to one week ahead for the Spanish system operator", it was noted that special days depend both on weekly and annual seasonal factors and a seasonal ARIMA model containing a weekly seasonal factor and yearly seasonal factor with dummy variables indicating special days and weather variables was constructed.

In the article "A comparison of univariate methods for forecasting electricity demand up to a day ahead" by Taylor, Menezes and McSharry (2006), Holt-Winter's Exponential Smoothing methodology has been extended in order to accommodate both intraweek and intraday seasonality on hourly time series and they called this method as exponential smoothing for double seasonality. Double seasonal exponential smoothing methods were reported to result in more precise forecasts than seasonal ARIMA and neural network approaches with quite stable time series having a length of at least 2 years. They also used regression with PCA (Principle Components Analysis) in order to reduce the dimensionality of multivariate data sets, where variables were highly correlated. In this method, they used six dummy variables indicating the day of the week, plus a linear and a quadratic term. Gould et al. (2008) used multiple seasonal patterns for forecasting hourly traffic flows in order to update one seasonal sub-cycle during the time for another sub-cycle.

Livera & Hyndman (2009) presented a new trigonometric formulation decomposing complex seasonal time series having multiple seasonal periods, high frequency and non-integer seasonality and dual (Gregorian and Hijri) calendar effects.

Hyndman et al. (2008), proposed a multi-seasonal exponential smoothing method for hourly time series. They divided a weekly seasonal cycle into daily seasonal cycles and assumed that days Monday-Friday have the same seasonal pattern, so they used the same sub cycle for these five days. Also, they used a second sub cycle which is the same for Saturday and Sunday. They reduced the number of the seasonal term estimates for hourly time series from 168 estimates per week for the HW (Holt Winter) method and 192 estimates for the DS (Double Seasonal) method to 48 estimates for their mentioned method.

In addition, most of the methods used in the articles do not aim at forecasting daily time series for lead times longer than one month, and special day effects and yearly seasonality in daily time series are not handled together. In Chapters 4, 5, 6, and 7 different daily time series forecasting methods providing longer term forecasts and additional processes for special day's effects and the change in sequential years' corresponding data level will be presented.

4. NAIVE METHOD

4.1. Naive Method

The naive method is one of the easiest methods used for forecasting. This method is based on the idea that the forecast for the next time period is equal to the last observed value or the value obtained by addition of a specific percentage of the last observed value to itself. The mathematical equation of the naive method is:

$$\widehat{y_t}(t-1) = y_{t-1} \tag{4.1}$$

In this equation, $\widehat{y_t}$ is the forecast value for period t and y_{t-1} is the observed value for period $t-1$.

The naive forecast method is generally used for time series which do not have strong local fluctuations. For multi-step-ahead forecasting of seasonal data, this method performs poorly.

4.2. Naive Method for Seasonal Daily Time Series

For seasonal data, a forecast of the period can be obtained by taking the value of the corresponding period observed in the last seasonal cycle (Taylor et al., 2006). The naive forecasting procedure is applied by Conejo et al. (2005) by taking the hourly values of the corresponding days of the last week for Monday, Saturday and Sunday and by taking the previous days' value for the other days of the week. By using this procedure at most seven-days-ahead forecasts can be obtained, whereas in many situations longer forecast horizons are required.

Seasonal daily time series which have yearly periodicity need a different variant of the naive method. In order to forecast values for longer horizons such as more than 3 months ahead, the yearly seasonality with periodicity of 364 or 365 days must be considered. If there is also weekly seasonality nested in yearly seasonality, the usage of the

frequency 364 is more reasonable since 364 is a multiple of 7, the number of weekdays. In addition to this, forecasting longer horizons than one or two weeks may require a "change ratio" identifying a change compared to 364 days before values. In my naive method applications, daily forecasts up to 3.5 months ahead were derived. I use the value 364 days before the forecast values and multiply it by the ratio of the most recent few days' sum in the history data to the sum of the same period last year. This can be expressed by Equation (4.2).

$$\widehat{y_t}(t-n) = y_{t-364} \frac{\sum_{i=0}^{k-1} y_{t-n-i}}{\sum_{i=0}^{k-1} y_{t-n-i-364}} \tag{4.2}$$

Here, k is the length in days of the time window used for obtaining the "change ratio". The notation $\widehat{y_t}(t-n)$ in Equation (4.2) represents that by observing the time series up to the time $t-n$, value of the time t is forecasted. In the next chapters, the name "naive method" is used for this method.

In addition, the period length of 364 days causes shifts and larger forecast errors for national holidays and adjacent days, since the yearly periodicity is in truth 365 days in general and 366 days in leap years. In order to prevent these shifts in special days an additional procedure is carried out. If the forecast value at time t is a special day value, in other words, if we are forecasting the value in a special day, then we do not take the value 364 days before the forecasted time t as we apply for normal days. But instead, we take the values at the date, whose year is one less than the year of the forecasted value and whose day and month is the same with the day and month of the forecasted value. For instance, if we are forecasting the value on December 31, 2010, which is a Christmas Holiday, then we do not take the value on a date 364 days before the forecasted day. We take the value on December 31, 2009. This means that if the forecasted day is in a year having the length of 365 days then we take the value 365 days before, or, if the forecasted day is in a year having the length of 366 days then we take the value 366 days before the forecasted day.

Similarly, 364 days before of a forecasted day can be a special day and the forecasted day can be a normal day. In this situation, we do not take the value 364 days before the forecasted value; instead we follow the same procedure explained in the previous

paragraph. For instance, if we are forecasting the value on December 23, 2010, which is not a special day, 364 days before of this day is December 24, 2009, since the year 2010 has a length of 365 days. The date of December 24, 2009 is a special day, which is Christmas Eve. So we take the value on the date of December 23, 2009 in order to multiply with the "change ratio".

It is likely that the forecast values for a specific time period will not be related solely to the data 364 days before but also will be related to the data 357, 371 days or similar days before. The reason for taking the values one week before and one week after the 364 days before the forecasted values is that the values 364 days before the forecasted values may be related to the time series values one week before and one week after. This variant of the naive forecast is described by Equation (4.3).

$$\widehat{y_t}(t-n) = (ay_{t-357} + by_{t-371} + cy_{t-364}) \frac{\sum_{i=0}^{k-1} y_{t-n-i}}{\sum_{i=0}^{k-1} y_{t-n-i-364}} \tag{4.3}$$

The values of these weights (or parameters) can be selected by minimizing the mean square of errors for a given fitting horizon with history data. This name "naive weighted average" is used for this variant of the naive method. The naive methods give comparatively good results with at least two years of data as we will see in our forecasting experiments.

In order to illustrate how the naive special day model performs disposing of special days' effect, forecast plot from the date of the December 10, 2008 to March 17, 2009 can be given as an example. The time series data shown in Figures 4.1 and 4.2 has a strong yearly seasonality and is stable. Also there is an obvious weekly seasonality nested in yearly seasonality.

In Figure 4.1, forecast values from the December 23, 2008 until the January 2, 2009 do not reveal any shifting from the observed values in the naive special day model. However, the naive model in Figure 4.2 which does not take into consideration the special days' effect reveals a shift at the end of December and at the beginning of January. The normal days are forecasted similarly in naive and naive special day model.

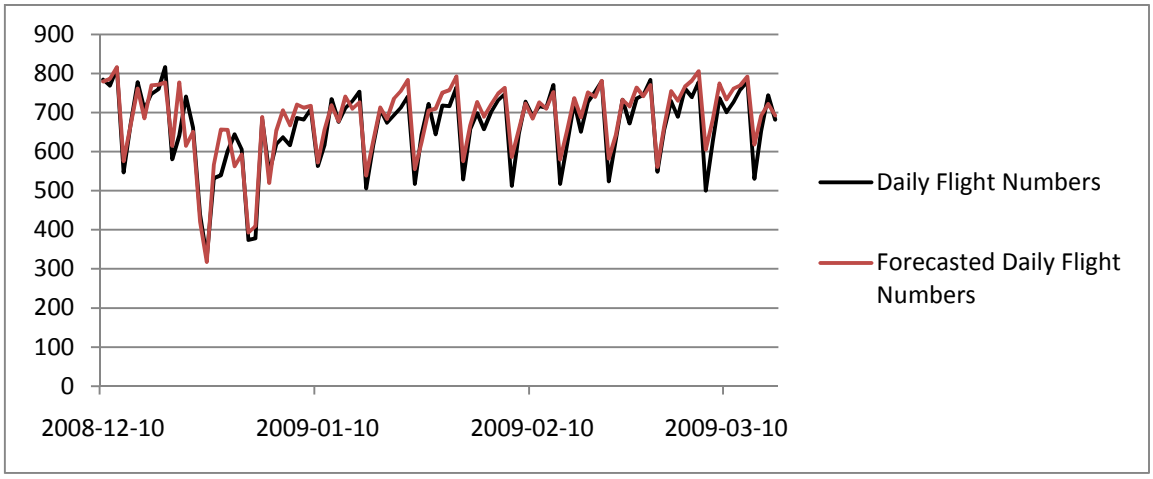

Figure 4.1. Forecast plot for Naive special day method

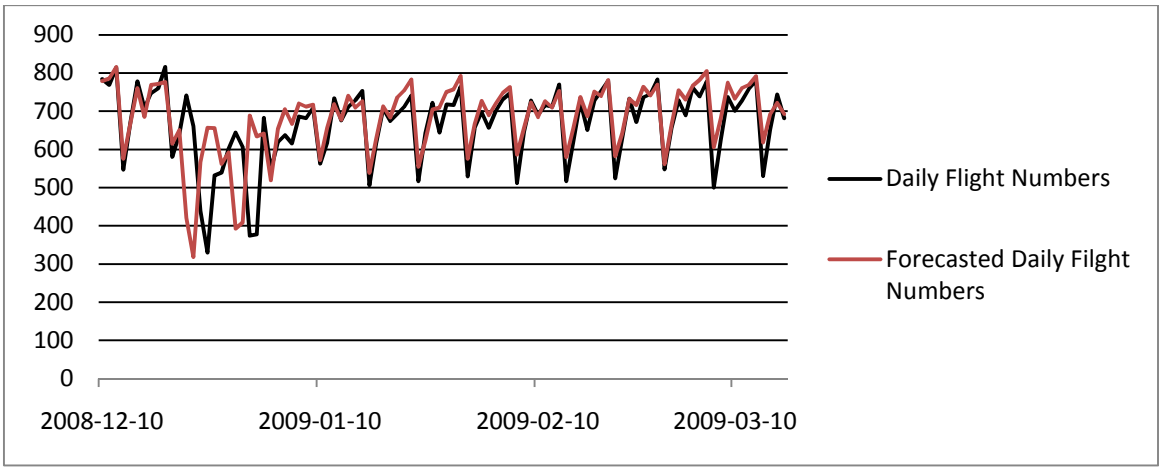

Figure 4.2. Forecast plot for Naive method

5. REGRESSION METHOD

5.1. Regression Method for Time Series Data

We are concerned with time series data here. The results of regression depend heavily on the assumption that errors are independent and predictor variables are not related to each other. But for time series data, the observations cannot be regarded as a random sample (Hanke & Wichern, 2009). When consecutive observations over time are related to one another, autocorrelation between observations exists. If values of a response in one time period are related to the values in previous time periods, then the previous periods' values can be used to predict future values (Hanke & Wichern, 2009). In this part, for the notations in equations, the book "Business Forecasting" is going to be used.

For instance, if there is first order serial autocorrelation in the error terms in the time series, then the error term in the current time period is directly related to the error term in the previous time period. In this situation simple linear regression can be written as:

$$Y_t = \beta_0 + \beta_1 X_t + \varepsilon_t \tag{5.1}$$

$$\varepsilon_t = \rho \varepsilon_{t-1} + v_t \tag{5.2}$$

As Hanke & Wichern (2009) state, if there is a correlation between adjacent error terms, then it is expected that there may be negative errors and positive errors sequentially located along the regression line. In this situation using the ordinary least squares approach to generate forecasts of future Y's is not appropriate. Also, a strong autocorrelation between errors can make two unrelated variables appear to be related and the standard regression procedures applied to the observations on these variables can produce a significant regression model. In this situation, the relationship is spurious and residual plots should be examined carefully.

5.2. Elimination of Autocorrelation in Time Series Data

For the time series data, there are some methods applied for avoiding autocorrelation between error terms. Hanke and Wichern (2009, p.347) state that after autocorrelation has been discovered in a regression of time series data, it is necessary to remove it or model it, before the regression function can be evaluated for its effectiveness. It is also stated that autocorrelation can arise because of a specification error such as omitted variable, or it can happen because of the correlated error terms. Since the main cause of auto correlated errors in the regression model is not to take into consideration one or more important variables, the possible solution to this problem can be to find them and include these variables in the regression. Another solution for eliminating autocorrelation is "differencing" the time series data. By differencing, the regression model is specified in terms of the changes in rather than levels of the series. Besides using first differences, generalized differences may also be required to avoid autocorrelation.

The method used for eliminating the autocorrelation generates predictor variables by using the response variable, Y, lagged one or more periods. The forecasting models which use lagged values of one or more independent or dependent variables are called distributed lag models since the influence of a variable is distributed over several time periods (Saunders et al., 1987). The expected value of the dependent variable at the current time period is influenced by the values taken by the independent or dependent variable at the current and previous time periods.

In the first order autoregressive model case, the only predictor variable is the Y variable lagged by one time period. The first order autoregressive model can be expressed as:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \varepsilon_t \tag{5.3}$$

In this equation, the errors ε_t are assumed to have the usual regression properties. In other words the error terms are expected to be normally and independently distributed with zero means. When this model is fit to the data by the ordinary least squares method, the one-period-ahead forecasting equation is stated as:

$$\hat{Y}_t = b_0 + b_1 Y_{t-1} \tag{5.4}$$

In this thesis, the regression models including autoregressive variables lagged by more than one time period are going to be used in order to eliminate the autocorrelation between error terms.

In addition, the inclusion of lagged values of the dependent variable on the right hand side of a regression equation brings about some estimation problems. When a lagged dependent variable is added in the regression, the error term enters the model in the usual form and it can be assumed that the error terms are normally and independently distributed with mean zero and constant variance. But, the presence of lagged dependent variables as explanatory variables violates the classical assumption that the explanatory variables are distributed independently of each other. Also, the consecutive values in a variable are correlated and this violates the classical regression assumptions. In time series, if the number of lags is large, then OLS (ordinary least squares) will give biased estimates in small samples. But using a sufficient number of observations reduces the biased estimates. In experimental part of this thesis, a large number of observations are used in the lagged regression models in order to accurately evaluate the significances of the variables.

5.3. Heteroscedasticity in Time Series Data

If random variables have different variances in different observations, then it can be interpreted that the random variables are heteroscedastic. Some time series' variability increases with the level of the series. So, non constant variability is called heteroscedasticity. In a regression, heteroscedasticity occurs if the variance of the error term, ε, is not constant. Hanke and Wichern (2009) states that if the variability for recent time periods is larger than it was for past time periods, then the standard error of the estimate underestimates the current standard deviation of the error term. The problem of heteroscedasticity observed in time series data can be solved by transformations of the data as mentioned in previous chapters. In this thesis, increase of the variability with the level increase in time series data is tried to be eliminated by logarithmic transformation of the time series data.

5.4. Additional Applications in Regression Models

5.4.1. Log-Linear Regression Models

Saunders et al. (1987) state that the primary transformation method for forecasting time series with regression models is the logarithmic transformation. As for many data, there is a multiplicative relationship rather than an additive relationship between the variables. The multiplicative relationship between the dependent and explanatory variables can be expressed as in Equation (5.5).

$$Y = aX_1^{b1}X_2^{b2} \dots X_k^{bk}e^{\varepsilon} \tag{5.5}$$

It is important to take into consideration that the error terms must also be treated as multiplicative rather than additive. When the natural logarithms of the variables are taken, a linear relationship between the variables is obtained. The natural logarithm of these variables can be expressed by Equation (5.6).

$$logY = loga + b_1logX_1 + b_2logX_2 + \cdots + b_klogX_k + \varepsilon \tag{5.6}$$

In this way, estimation of the parameters in this regression equation can be obtained by the ordinary least squares approach. We used logarithmic regression equations in this thesis in order to transform a multiplicative relationship between regression variables into an additive relationship.

5.4.2. Dummy Variables

In linear regression models dummy variables can also be included as explanatory variables (Saunders et al., 1987). These are binary variables which generally take on the value 0 or 1 and may be used to represent existence or non-existence of different situations. For example, dummy variables can be used to represent seasonal patterns, non quantifiable characteristics of variables, and other special characteristics of variables which affect the value of dependent variable in regression. The purpose of using dummy variables in some

regression models developed in this thesis is that dummy variables help to indicate the special days such as national holidays since there is an observable change in the values on special days.

5.5. Regression Models for Daily Time Series Studied

There are some papers using regression models for daily and hourly forecasting. Lemke & Gabrys (2010) used polynomial regression of orders between two and six by grid search and extrapolated future values for forecasting daily data, but their method did not result in a small MAPE when compared to other methods. Taylor (2006) proposed a regression model with principal components analysis indicating the days of the week as dummy variables with a linear and a quadratic term, which was outperformed by exponential smoothing methods. Weatherford & Kimes (2003) suggested a linear regression with reservations as the dependent variable against bookings a few days before, but exponential smoothing method outperformed the proposed linear and log linear regressions.

The application of the regression methods in this thesis is performed by the conjecture that there is a multiplicative and lagged relationship between regressors and dependent variables. This situation can be described by the formula:

$$\widehat{Y_t} = aR^{b_0} Y_{t-364}^{b_1} Y_{t-371}^{b_2} Y_{t-357}^{b_3} Y_{t-365}^{b_4} Y_{t-728}^{b_5} Y_{t-730}^{b_6} \tag{5.7}$$

Here R denotes the change ratio of the sum of the last observed values of the series over the sum of the values observed 364 days before, and a is the constant term of the regression. By taking the logarithm of the equation we obtain:

$$\widehat{logY_t} = loga + b_0 logR + b_1 logY_{t-364} + b_2 logY_{t-371} + b_3 logY_{t-357} + b_4 logY_{t-365} + b_5 logY_{t-728} + b_6 logY_{t-730} \tag{5.8}$$

In the Chapter 4, the *naive weighted average* method is also a regression method, but it assumes that there is an additive relationship between lagged variables. In the *regression*

method, a (0,1) dummy variable for special and normal days is not used and some of the regression variables are not added to the model, as they are not significant. The *regspecial_all* regression model includes all variables and dummy variables designating special days and normal days. *Dummy* variables are used as an indicator of special days and are not used directly in the regression equation. The *regspecial_remove_insignif* regression model uses dummy variables but does not use insignificant variables of the regression equation and gets the regression results according to remaining significant variables. The *regspecial_x364* model assumes that only the lagged variableY_{t-364}, the R ratio variable and dummy variables are necessary for satisfactory forecasting results. The models *regspecial_all*, *regspecial_remove_insignif* and *regspecial_x364* are log linear and autoregressive regression models.

Residual and Normal Q-Q plots belonging to one of the historic forecast for normal days in the *regspecial_all* regression model are given in the Figures 5.1 and 5.2. The reason for analysing these plots is to observe if the variance of the observed time series values increase with the level of the time series and if the residuals are distributed normally and independently. It can be seen that most of the standardized residuals lie in the interval (-3,3) and the Normal Q-Q plot in the Figure 5.2 appears satisfactory but the residuals plotted in Figure 5.1 are not very homogenous.

For the *regspecial_all* model ACF of residuals is plotted in Figure 5.3. There are some positive and then negative autocorrelations between residuals for a brief duration, so the residuals are not independently distributed. But the autocorrelation between residuals are not significant after a few lags. It can be seen that the standard regression model assumptions are not satisfied perfectly and the auto correlated errors problem that we explained in part 5.2 cannot be solved completely.

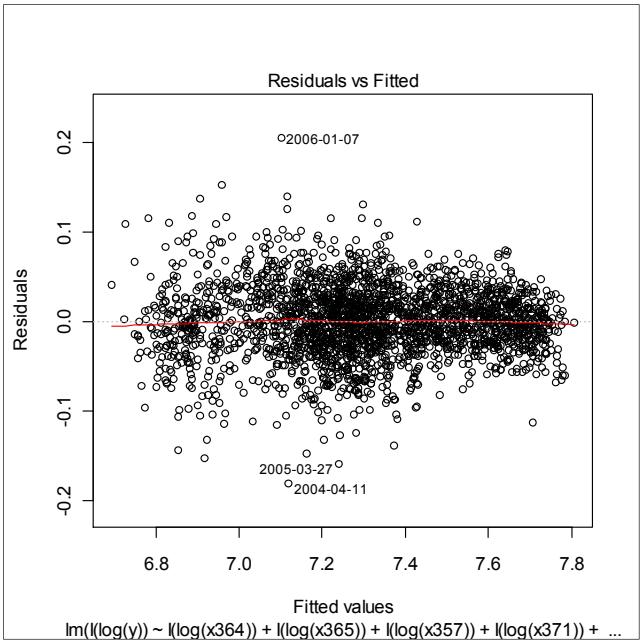

Figure 5.1. Residuals vs Fitted plot for the Regspecial_all method

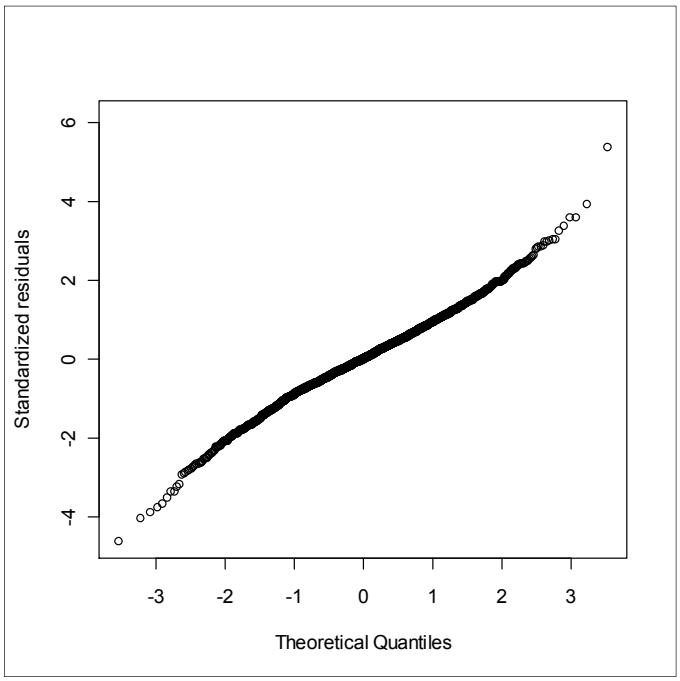

Figure 5.2. Normal Q-Q plot for Regspecial_all method

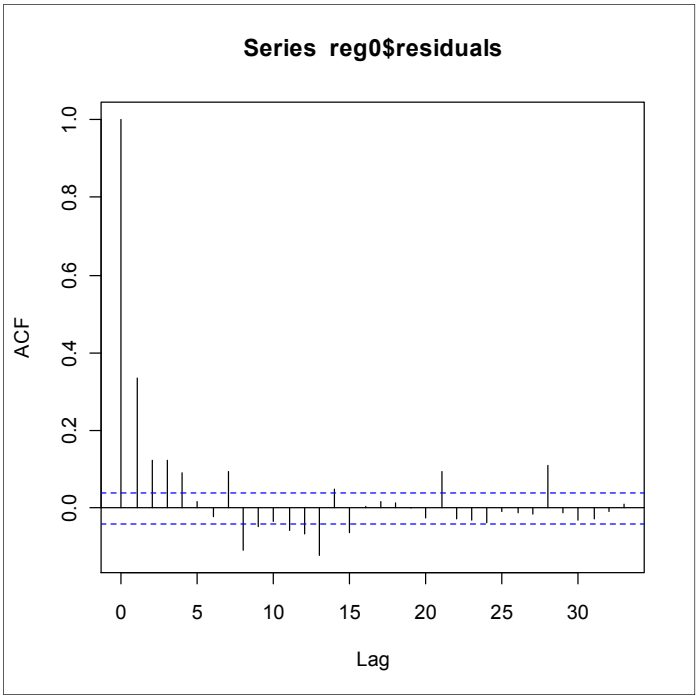

Figure 5.3. ACF plot for Regspecial_all method residuals

Residuals plot belonging to *regression* model indicates that the residuals are not normally distributed since most of them do not lie between (-3,3) interval in Normal QQ plot. Also, it can be seen that *regression* model has more outliers than *regspecial_all* model since, the special days and the normal days are included in the same regression equation without a dummy variable indicating the special days. In the *regression* model, special days' effect is not taken into consideration, but the shift in special days seasonality is accounted for by using dependent variables lagged one week less (364 days) and one week more (371 days) than the 364-day-lagged variable.

The extreme outliers are associated with the Christmas Holiday. In the *regression* method the standard model assumptions are not fulfilled as in the previous *regspecial_all* model.

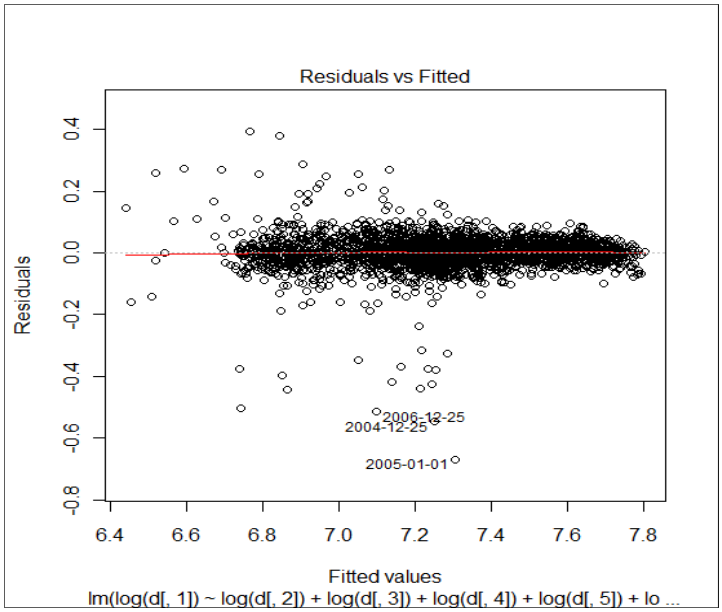

Figure 5.4. Residuals vs. fitted plot for the Regression method

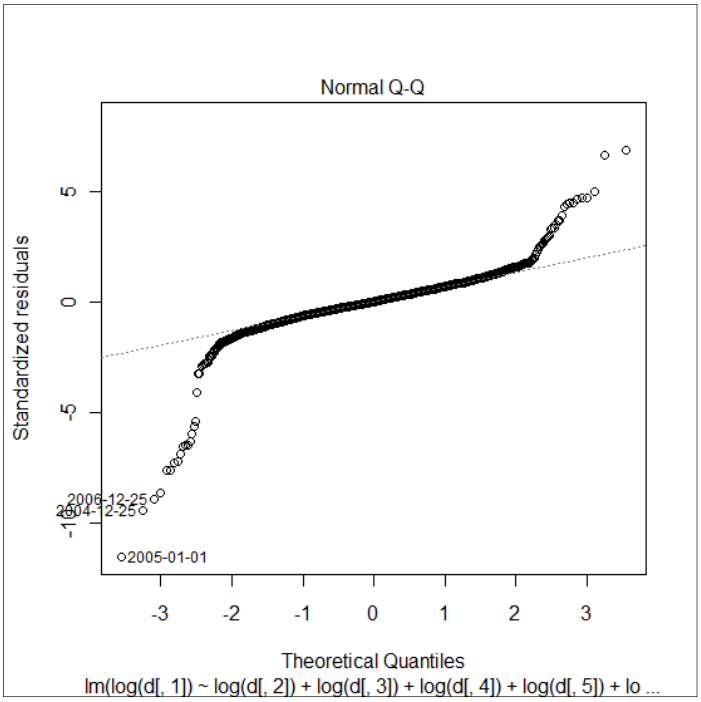

Figure 5.5. Normal Q-Q plot for the Regression method

In the previous chapter, the method “naive weighted average” is also a regression method as stated before. The last part of the data having a length of nearly one year is cut from time series and for this part lagged variables are used in order to determine optimum

parameters for fitting process. In this method there is an assumption that the determined parameters for the last cut part of the data can also explain the relationship between later used lagged variables in the forecasting period. As, this method is an additive regression rather than a multiplicative regression, it is included in the “naive method” part. In the following chapters, more detailed explanations of methods are going to be given via the explanations of “R” functions.

6. EXPONENTIAL SMOOTHING METHODS

This approach assumes that time series data consist of two components of signal and noise. Smoothing can be seen as a technique to separate the signal and the noise and in that a smoother acts as a filter to obtain an "estimate" for the signal (Montgomery et al., 2008). The signal represents any pattern caused by the intrinsic dynamics of the process. This pattern can be a simple constant process or a more complicated structure that cannot be extracted visually or with statistical tools. The different types of exponential smoothing methods used in forecasting time series data are explained in this chapter.

6.1. Simple Exponential Smoothing

While the method of moving averages takes into account only the most recent observations, simple exponential smoothing provides an exponentially weighted moving average of all previously observed values. The model is often appropriate for data with no predictable upward or downward trend. The aim is to estimate the current level. Simple exponential smoothing in recursive form can be described by:

$$\hat{y}_T = \alpha y_T + (1-\alpha)\alpha(y_{T-1} + (1-\alpha)y_{T-2} + (1-\alpha)^2 y_{T-3} + \cdots + (1-\alpha)^{T-2} y_1) \tag{6.1}$$

$$= \alpha y_T + (1-\alpha)\hat{y}_{T-1} \tag{6.2}$$

In Equations (6.1) and (6.2), we need initialization for the first estimation since a linear combination of the current observation and the smoothed observation at the previous time period is used for all estimation. The estimate of $\hat{y}_0$ has little relevance for any time series (Montgomery, 2008).

6.2. Double Exponential Smoothing

In simple exponential smoothing, the level of the time series is assumed to be changing occasionally and an estimate of the current level is required. In some situations, the observed data will be clearly trending and contain information that allows anticipation of future upward or downward movements. When a trend in the time series is anticipated, an estimate of the future slope, with the future level, is required.

The three equations used in double exponential smoothing which also called "Holt's linear exponential smoothing" are:

The level estimate

$$L_t = \alpha Y_t + (1 - \alpha)(L_{t-1} + T_{t-1}) \tag{6.3}$$

The trend estimate

$$T_t = \beta(L_t - L_{t-1}) + (1 - \beta)T_{t-1} \tag{6.4}$$

The forecast n periods into the future

$$\hat{Y}_{t+n}(t) = L_t + nT_t \tag{6.5}$$

Later, we will use Holt's method in our experiments.

6.3. Holt Winter's Seasonal Exponential Smoothing

Many time series data feature cyclical or seasonal patterns that cannot be effectively modelled using a polynomial model. Winter's three parameter linear and seasonal exponential smoothing method, an extension of Holt's method, often represents the data better and reduces forecast error. An important point to note regarding the seasonal

exponential smoothing approach is that in contrast with SARIMA (seasonal ARIMA) modelling, there is no model specification involved. The two commonly used Holt-Winters exponential smoothing models are multiplicative and additive seasonal models. We should decide about the appropriate model according to the statistical properties of the time series to forecast. If the seasonal variation increases with the increase in the mean of the series, then using the multiplicative form of the model is more suitable.

The four equations used in Winter's (multiplicative) smoothing are:

The level estimate

$$L_t = \alpha \frac{Y_t}{S_{t-s}} + (1-\alpha)(L_{t-1} + T_{t-1}) \tag{6.6}$$

The trend estimate

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \tag{6.7}$$

The seasonality estimate

$$S_t = \gamma \frac{Y_t}{L_t} + (1-\gamma)S_{t-s} \tag{6.8}$$

The forecast n periods into the future

$$\hat{Y}_{t+n}(t) = (L_t + nT_t)S_{t-s+n} \tag{6.9}$$

The four equations used in Winter's (additive) smoothing are:

The level estimate

$$L_t = \alpha(Y_t - S_{t-s}) + (1-\alpha)(L_{t-1} + T_{t-1}) \tag{6.10}$$

The trend estimate

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \tag{6.11}$$

The seasonality estimate

$$S_t = \gamma(Y_t - L_t) + (1-\gamma)S_{t-s} \tag{6.12}$$

The forecast n periods into the future

$$\hat{Y}_{t+n}(t) = L_t + nT_t + S_{t-s+n} \tag{6.13}$$

Later, we will use both variants of these seasonal exponential smoothing methods.

6.4. Double Seasonal Exponential Smoothing

Taylor et al. (2006) state that the application of double seasonal exponential smoothing requires an extension of the standard Holt-Winter's exponential smoothing formulation to accommodate two seasonal cycles in time series. This involves the introduction of an additional seasonal index and an extra smoothing equation for it. In their article, the authors have modelled double seasonality in electricity consumption for forecasting consumption up to a day ahead. The forecasted hours are related to the same hour of the previous day and to the same hour of the same day of the previous week. The formulation for double multiplicative seasonality additive trend exponential smoothing is given by the following expressions:

$$L_t = \alpha\left(\frac{y_t}{D_{t-s1}W_{t-s2}}\right) + (1-\alpha)(L_{t-1} + T_{t-1}) \tag{6.14}$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \tag{6.15}$$

$$D_t = \gamma\left(\frac{y_t}{L_t W_{t-s2}}\right) + (1-\gamma)D_{t-s1} \tag{6.16}$$

$$W_t = \omega\left(\frac{y_t}{L_t D_{t-s1}}\right) + (1-\omega)W_{t-s2} \tag{6.17}$$

$$\hat{Y}_{t+n}(t) = (L_t + nT_t)D_{t-s1+n}W_{t-s2+n} + \varphi^n(y_t - \left((L_{t-1} + T_{t-1})D_{t-s1}W_{t-s2}\right) \tag{6.18}$$

L_t and T_t are the smoothed level and trend; D_t and W_t are the smoothed seasonal indices for the two intraday and intraweek seasonal cycles, respectively Taylor et al. (2006). α, β, γ and ω are the smoothing parameters; and $\hat{Y}_{t+n}(t)$ is the n-step-ahead forecast made as of time t. The term involving the parameter φ in the forecast function is a simple adjustment for the first order autocorrelation. All the parameters,α, β, γ, ω and φ are estimated in a single procedure by minimizing the sum of squared model fit errors. The initial smoothed values for the level, trend and seasonal components are estimated by averaging the early observations.

Gould et al. (2008) cited by Taylor (2010) note that the double seasonal Holt-Winter's method assumed that any intraday cycle was the same for the seven days of a week. Gould et al. thus considered intraday cycles different for every weekday, although they did not mention any intraweek seasonality.

In our applications for daily time series the double seasonal exponential smoothing method is used, and intraweek and intrayear seasonality are used instead of intraday and intraweek seasonality, such that the period of weekly seasonality has length 7 and the yearly seasonality has length 364. The weekly seasonality is postulated to change every each year and stay the same for a year period in the adjustment part of seasonal indices and parameters, which are going to be used then for the forecasted values. Since we are going to simulate the model by including the new value in every day, this adjustment period and so parameters are updated every time when we include the recent day value.

For long termed forecasts, if the weekly seasonal indices are different for the different periods of a year, double seasonal exponential smoothing methods are not expected to have good forecast performance, since it uses a yearly seasonal index with the

repeating weekly seasonal indices in a whole year. The weekly seasonal indices term indicates “daily indices of weekly seasonality”. When yearly seasonality is multiplicative, then weekly seasonality is taken as multiplicative and when yearly seasonality is additive, then weekly seasonality is taken as additive. Also, in order to test the autocorrelation term’s significance, we try double seasonal models with and without the parameter φ. In previous researches, the double seasonal exponential smoothing method was applied for hourly time series data, but in our experiments, daily time series data is used.

Figures 6.1 and 6.2 sequentially illustrate weekly (intraweek) and yearly (intrayear) seasonality indices of *double multiplicative seasonal and additive trend exponential smoothing (doublemsat)* model along a horizon of two years. The horizontal axis shows days.

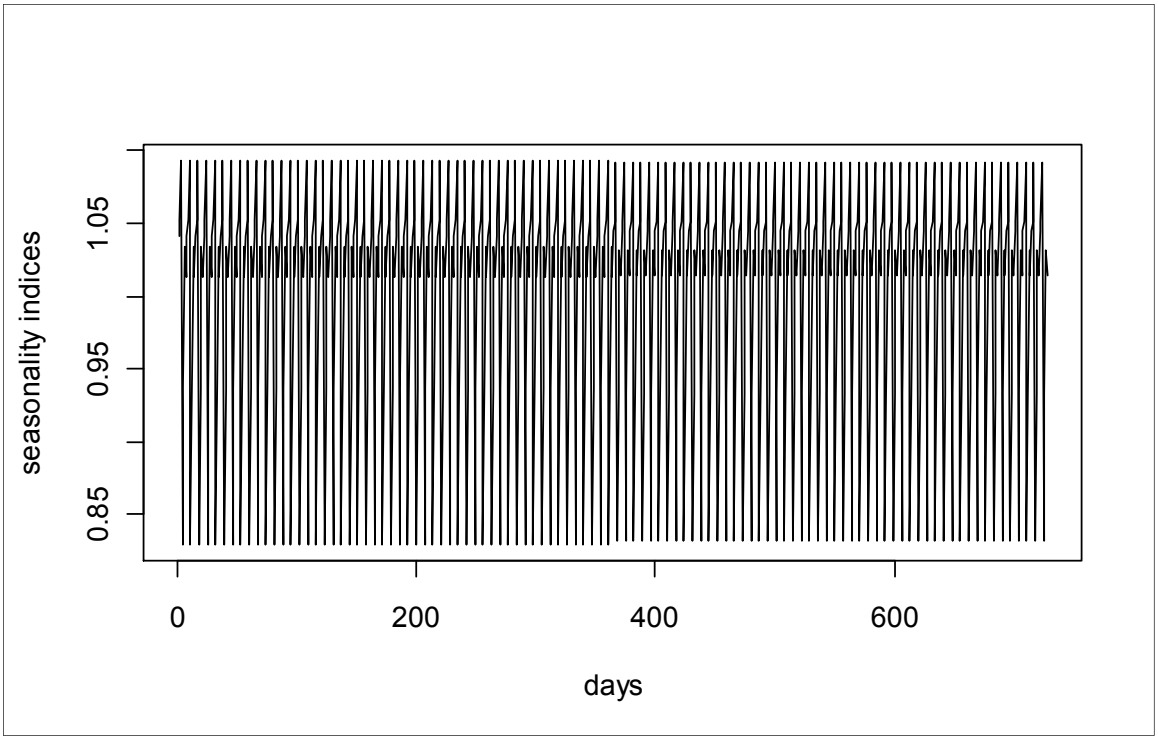

Figure 6.1. Weekly Seasonal Indices of the Double Multiplicative Seasonal and Additive Trend Exponential Smoothing Method

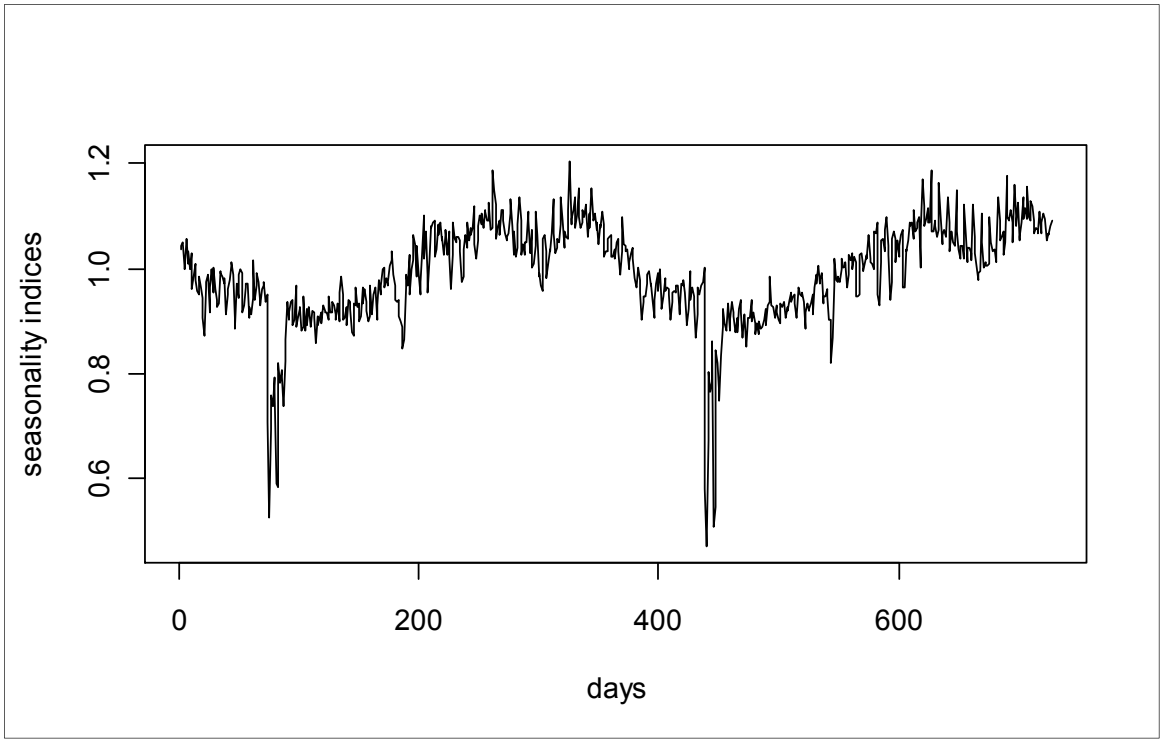

Figure 6.2. Yearly Seasonal Indices of the Double Multiplicative Seasonal and Additive Trend Exponential Smoothing Method

6.5. Multiple Seasonal Exponential Smoothing

In most of the papers, multiple seasonality has been used for hourly time series. According to Taylor's (2010) double seasonal method, the same intraday cycle is repeated for all days of the week. In addition, the updates of seasonal indices based upon recent information are the same for each day of the week. However, the intraday cycles in a week, or intaweek cycles in a year may differ in all days of a week or in all weeks of a year. In the article of Gould et al. (2008), seasonal components can be updated more frequently than once during a seasonal cycle.

Gould et al. (2008) divided the cycle of length m_2 into k shorter sub-cycles of length m_1. They suggested a matrix of smoothing parameters that allows the seasonal terms of one sub-cycle to be updated during the time for another sub-cycle. For instance, seasonal terms for a week can be updated in the next week. Also, this goal can be achieved by combining the sub-cycles with the same seasonal pattern into one common sub-cycle. This

approach has the advantage of reducing the number of seed values that are needed with the multiple seasonal models. It is also possible to have different smoothing parameters for different sub-cycles (for different days of the week).

In the empirical part of this thesis, multiple seasonal exponential smoothing methods are used in a different way. Since studied time series data are daily data and the purpose is not to forecast just up to a week ahead, some modifications have to be used for the described multiple seasonal models. If the same model stated above had been applied, then the meaningful forecast horizon could be at most one week. The reason of it can be explained that, while forecasting the values in a week, the previous week's seasonal indices should be updated. So, when we want to forecast up to six weeks, for example, we cannot update the fifth week's weekly seasonal indices even when we do not know the existing weekly seasonal indices of it. The sub-cycle length determines the meaningful forecast horizon unless we modify the existing model. Since the required forecast horizon is up to 14 weeks ahead, and the 14-week-ahead forecasts cannot use the weekly seasonal indices 13 weeks ahead, then weekly seasonal indices are updated every week for two years and different weekly seasonal indices are obtained during a year.

Yearly and weekly seasonal indices that were available 364 days earlier than the day to forecast could be used in order to generate forecasts over longer horizons. In addition, in order to obtain more accurate results, weekly sub-cycles such as Saturday to Sunday and Monday to Friday are not combined. In this method, it is assumed that weekly seasonal indices differ within a year, but for consecutive years, the same week's weekly seasonality is comparable. Nevertheless, double seasonal exponential smoothing is expected to give good forecast results when there is no significant weekly seasonality difference between different periods of a year.

In the multiple seasonal exponential smoothing methods, daily indices of weekly seasonality is not the same in every week and daily indices of weekly seasonality are updated every week along two years in order to be able to use them in forecasting periods. Yearly seasonal frequency is taken as 364 days and weekly seasonal frequency is taken as 7 days as in the double seasonal exponential smoothing method. Weekly and yearly seasonality smoothing parameters are optimized in order to give the minimum MSE of

one-step-ahead forecasts for a two year period for the time series data having sufficient length. Once a two year period of one step ahead forecast parameters are optimized, then forecast values up to the forecast horizon are generated using the optimized parameters.

The prediction of a day is assumed to be related to the previous year's yearly seasonal indices and the previous year's weekly seasonal indices, and for every prediction period, the weekly seasonal indices change. Then, the updated weekly and yearly seasonal indices can be used to predict up to 364 days ahead. However for our experiments, we forecasted up to 14 weeks ahead. Figures 6.3 and 6.4 illustrate the weekly and yearly seasonality indices for *multiple multiplicative seasonal and additive trend exponential smoothing* models. Horizontal axis shows days. Data used here has a strong yearly seasonality and stable. It can be seen that the weekly seasonal indices are different for every week of the year.

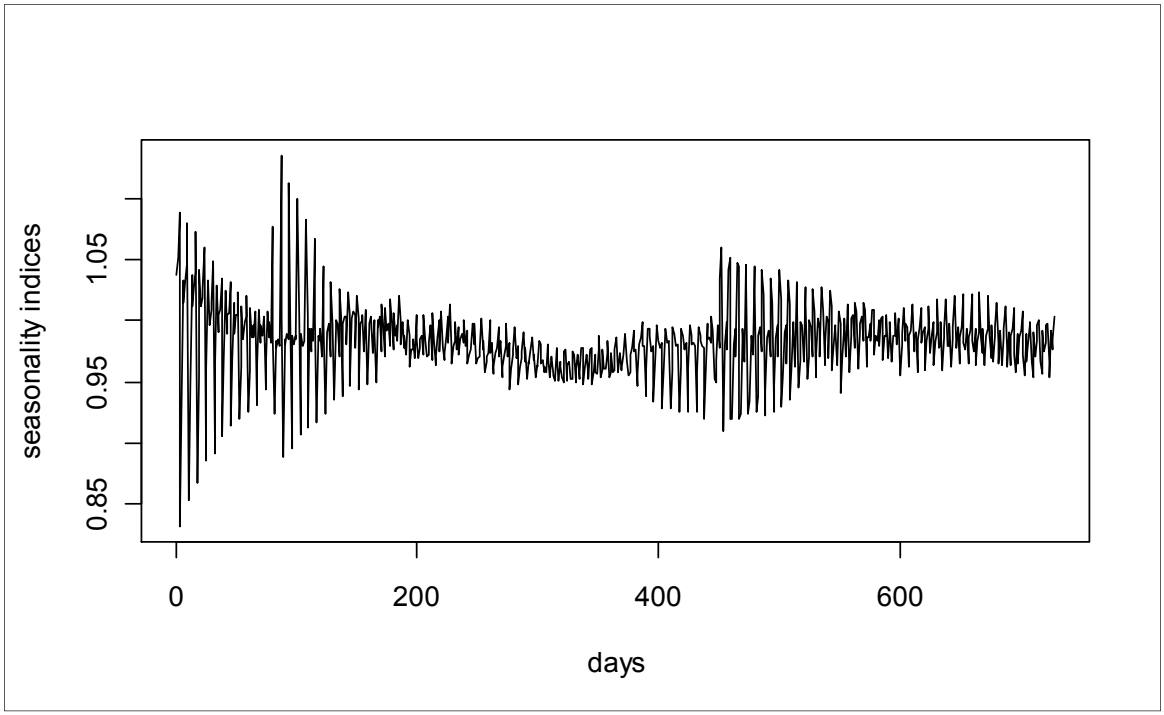

Figure 6.3. Weekly Seasonal Indices of the Multiple Multiplicative Seasonal and Additive Trend Exponential Smoothing Method

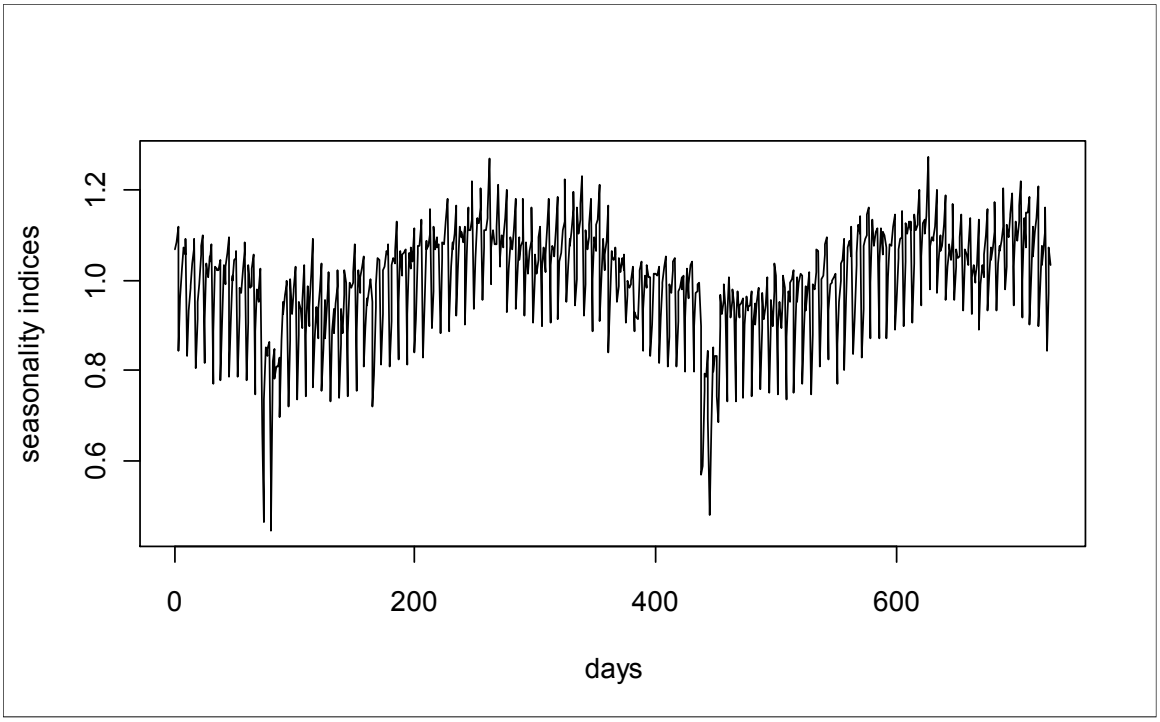

Figure 6.4. Yearly Seasonal Indices of the Multiple Multiplicative Seasonal and Additive Trend Exponential Smoothing Method

6.6. Multiplicative and Additive Damped Trend in Exponential Smoothing

In general, an additive trend is used in exponential smoothing methods. However, it has been noted that more real series have multiplicative trends than additive. Taylor (2003) states that modelling trends in a multiplicative form by including the dampening term results in more robust forecasting performance. Hyndman et al. (2002) has included a damped additive trend to non seasonal, additive seasonal and multiplicative seasonal exponential smoothing methods.

In Winter's multiplicative seasonal model with a damped additive trend, a dampening parameter θ is multiplied by the trend component.

The level estimate

$$L_t = \alpha \frac{Y_t}{S_{t-s}} + (1-\alpha)(L_{t-1} + \theta T_{t-1}) \tag{6.19}$$

The trend estimate

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)\theta T_{t-1} \tag{6.20}$$

The seasonality estimate

$$S_t = \gamma \frac{Y_t}{L_t} + (1-\gamma)S_{t-s} \tag{6.21}$$

The forecast for n periods into the future

$$\hat{Y}_{t+n}(t) = (L_t + \sum_{i=1}^{n} \theta^i T_t)S_{t-s+n} \tag{6.22}$$

Winter's additive seasonal model with a damped additive trend is described by:

The level estimate

$$L_t = \alpha(Y_t - S_{t-s}) + (1-\alpha)(L_{t-1} + \theta T_{t-1}) \tag{6.23}$$

The trend estimate

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)\theta T_{t-1} \tag{6.24}$$

The seasonality estimate

$$S_t = \gamma(Y_t - L_t) + (1-\gamma)S_{t-s} \tag{6.25}$$

The forecast for n periods into the future

$$\hat{Y}_{t+n}(t) = L_t + \sum_{i=1}^{n} \theta^i T_t + S_{t-s+n} \tag{6.26}$$

When the time series have an exponential trend, Pegels has suggested a multiplicative trend exponential smoothing method (Taylor, 2003). The multiplicative trend can be used in Winter's additive and multiplicative seasonal exponential smoothing methods. The methods shown below models the trend in a multiplicative way and forecasts are obtained by level, growth rates and seasonality terms.

Winter's multiplicative seasonal model with a damped Pegels' multiplicative trend is presented in expressions:

The level estimate

$$L_t = \alpha \frac{Y_t}{S_{t-s}} + (1-\alpha)(L_{t-1}R_{t-1}^{\theta}) \tag{6.27}$$

The trend estimate

$$R_t = \beta(L_t/L_{t-1}) + (1-\beta)R_{t-1}^{\theta} \tag{6.28}$$

The seasonality estimate

$$S_t = \gamma \frac{Y_t}{L_t} + (1-\gamma)S_{t-s} \tag{6.29}$$

The forecast for n periods into the future

$$\hat{Y}_{t+n}(t) = L_t R_t^{\sum_{i=1}^{n}\theta^i} S_{t-s+n} \tag{6.30}$$

Here, R_t is the growth rate.

Winter's additive seasonal model with a damped Pegels' multiplicative trend is described by:

The level estimate

$$L_t = \alpha(Y_t - S_{t-s}) + (1-\alpha)(L_{t-1}R_{t-1}^{\theta}) \tag{6.31}$$

The trend estimate

$$T_t = \beta(L_t/L_{t-1}) + (1-\beta)R_{t-1}^{\theta} \tag{6.32}$$

The seasonality estimate

$$S_t = \gamma(Y_t - L_t) + (1-\gamma)S_{t-s} \tag{6.33}$$

The forecast for n periods into the future

$$\hat{Y}_{t+n}(t) = L_t R_t^{\sum_{i=1}^{n}\theta^i} + S_{t-s+n} \tag{6.34}$$

Taylor (2003) argues that a logarithmic transformation is sometimes used to convert a multiplicative trend into an additive trend. The forecast results must then be transformed back into original units, along with prediction intervals. Using Pegels' method in seasonal exponential smoothing models also avoids such data transformation. It should also be noted that there is no equivalent ARIMA model for either the Pegels' or the new damped Pegels' methods. Grubb and Mason (2001) claim that dampening the trend to an average of some values improves the performance at most forecast horizons.

The initialization of the parameters used in exponential smoothing is important as the other parameters are calculated using the initial parameters. In additive trend exponential smoothing models, a simple linear regression is applied to the deseasonalized data in order to find the level and trend components. The number of periods used for the initialization should be at least two, in order to find the initial trend and level. For the damped Pegels' method the same initial level is used, but the initial growth rate R_0 equals to $\frac{(L_0+T_0)}{L_0}$. The

initial seasonal indices are calculated by multiplicative and additive decomposition of the first two years time series data. In the experiments here the multiplicative, damped multiplicative and additive trend methods are applied to single, double and multiple seasonal exponential smoothing methods in order to compare multiplicative and additive trend models.

6.7. Exponential Smoothing Methods for Daily Time Series Data

Exponential smoothing methods developed with different combinations of seasonal, trend and level components are used to compare the results over different data sets in order to find the exponential forecasting method giving the minimum mean squared forecast errors. In this way, a total of 28 exponential smoothing methods were obtained.

Table 6.1. Abbreviations used for Exponential Smoothing Methods

Exponential Smoothing Methods	**Abbreviations for Methods**
Double additive seasonal and additive trend	doubleasat
Double additive seasonal and additive trend with autocorrelation term	doubleasatauto
Double additive seasonal and damped multiplicative trend	doubleasdmt
Double additive seasonal and damped multiplicative trend with autocorrelation term	doubleasdmtauto
Double additive seasonal and multiplicative trend with autocorrelation term	doubleasmtauto
Double multiplicative seasonal and additive trend	doublemsat
Double multiplicative seasonal and additive trend with autocorrelation term	doublemsatauto
Double multiplicative seasonal and damped multiplicative trend	doublemsdmt
Double multiplicative seasonal and damped multiplicative trend with autocorrelation term	doublemsdmtauto
Double multiplicative seasonal and multiplicative trend with autocorrelation term	doublemsmtauto
Double exponential smoothing method	holtsexposmooth
Multiple additive seasonal and additive trend	multipleasat

Table 6.1. (continued)

Multiple additive seasonal and additive trend with autocorrelation term	multipleasatauto
Multiple additive seasonal and damped multiplicative trend	multipleasdmt
Multiple additive seasonal and damped multiplicative trend with autocorrelation term	multipleasdmtauto
Multiple additive seasonal and multiplicative trend with autocorrelation term	multipleasmtauto
Multiple multiplicative seasonal and additive trend	multiplemsat
Multiple multiplicative seasonal and additive trend with autocorrelation term	multiplemsatauto
Multiple multiplicative seasonal and damped multiplicative trend	multiplemsdmt
Multiple multiplicative seasonal and damped multiplicative trend with autocorrelation term	multiplemsdmtauto
Multiple multiplicative seasonal and multiplicative trend with autocorrelation term	multiplemsmtauto
Simple exponential smoothing	simpleexposmooth
Single additive seasonal and additive trend	singleasat
Single additive seasonal and damped multiplicative trend	singleasdmt
Single additive seasonal and multiplicative trend	singleasmt
Single multiplicative seasonal and additive trend	singlemsat
Single multiplicative seasonal and damped multiplicative trend	singlemsdmt
Single multiplicative seasonal and multiplicative trend	singlemsmt

In Chapter 8, the “R” functions performing the stated exponential smoothing models are explained. The model names are given to the functions coded in the “R” program.

7. BOX-JENKINS (ARIMA) METHOD

The Box-Jenkins methodology of forecasting is different from most methods because it does not assume any particular pattern in the historical data of the series to be forecasted. It uses an iterative approach of identifying a possible model from a general class of models .The chosen model is then checked against the historical data to see whether it accurately describes the series. The model fits well if the residuals are generally small and randomly distributed and contain no useful information. If the specified model is not satisfactory, the process is repeated using a new model designed to improve on the original one. This iterative procedure continues until a satisfactory model is found.

The initial selection of an ARIMA model is based on an examination of a plot of the time series in order to observe its general character and an examination of its autocorrelation for several time lags. Specifically, the autocorrelation function and partial autocorrelation function are used to select the order and season of the SARIMA model.

Hyndman and Khandakar (2008) state that in general, the order selection of the SARIMA and ARIMA models are subjective and difficult to apply. Another obstacle of using ARIMA models is that when new data are added, the order of the ARIMA models may need to be updated. Also, ARIMA models assume homoscedasticity. As a result, often data transformations are required for achieving stationary time series data for the model.

A p^{th} order autoregressive model takes the form:

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \varepsilon_t \tag{7.1}$$

Autoregressive models are appropriate for stationary time series, and the coefficient ϕ_0 is related to the constant level of the series. The autocorrelation coefficients trail off to zero gradually, whereas the partial autocorrelation coefficients drop to zero after the p^{th} time lag.

A q^{th} order moving average model takes the form:

$$Y_t = \mu + \varepsilon_t - \Delta_1\varepsilon_{t-1} - \Delta_2\varepsilon_{t-2} - \cdots - \Delta_q\varepsilon_{t-q} \quad (7.2)$$

Moving average (MA) models provide forecasts of Y_t using a linear combination of a finite number of past errors, whereas autoregressive (AR) models forecast Y_t as a linear function of a finite number of past values of Y_t. Moving average refers to the fact that the deviation of the response from its mean, $Y_t - \mu$, is a linear combination of current and past errors and that, as time moves forward, the errors involved in this linear combination move forward as well.

The autocorrelation coefficients for the MA (q) model drop to zero after the q^{th} time lag, whereas the partial autocorrelation coefficients trail off to zero gradually.

An ARMA (p,q) model has the general form

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \varepsilon_t - \Delta_1\varepsilon_{t-1} - \Delta_2\varepsilon_{t-2} - \cdots - \Delta_q\varepsilon_{t-q} \quad (7.3)$$

The ARMA (p,q) model depends on current and past values of the response Y, and the current and past values of errors.

Non-stationary time series models are called autoregressive integrated moving average models and denoted by ARIMA (p,d,q). Here, p indicates the order of the autoregressive part, d indicates the amount of differencing, and q indicates the order of the moving average part. Gardner and McKenzie (1985) (cited by Grubb and Mason, 2001) stated that *n*-period-ahead forecasts from (non-seasonal) ARIMA models have been found to perform poorly for large *n*.

7.1. Seasonal ARIMA Models

Seasonal ARIMA or SARIMA models contain regular autoregressive and moving average terms for the correlation at low lags and seasonal autoregressive and moving average terms for the correlation in the seasonal lags. In addition, for non stationary

seasonal series, seasonal differencing is often required to completely specify the model. The advantage of the SARIMA model over the ARIMA model is that it can deal with seasonality (Tong and Liang, 2002).

7.2. Seasonal ARIMA Models for Daily Time Series Data

There are several articles about forecasting daily and hourly time series, often to forecast electricity demand. The SARIMA electricity demand models mainly aim at forecasting up to a day or two days ahead. However, daily time series usually have yearly seasonality and there is a need for forecasting longer lead times. In previous applications on hourly ARIMA models, intraday and intraweek cycles were used in order to forecast future values. Taylor et al. (2006) have used a multiplicative double seasonal ARIMA model having an intraday cycle of 24 hours and an intraweek cycle of 168 hours in order to forecast electricity demand up to a day ahead. This means that Taylor related the prediction of any hour of the following day with 24 hours and 168 hours earlier data. Cancelo et al. (2008) have also used an ARIMA model with seasonality frequencies of 365 days and 7 days for one week ahead and one day ahead hourly predictions. Their study has included the usage of separate models for each hour of the day. It can be interpreted that while using a seasonal frequency of 365 days, weekly seasonality has also been included, and with the constraint of the smallest seasonal frequency, at most 7 days ahead predictions could be generated with updated weekly seasonality indices. Most of the SARIMA programs do not allow the usage of seasonality frequencies up to 364 or 365, so some difficulties appear in forecasting distant periods. A disadvantage of using SARIMA models is that order and seasonal terms may change when new data are added to existing data since characteristics of data may change. So, SARIMA models may need to be re-identified every few periods.

In our experiments, we used an approach which models each day of the week as separate time series with a seasonality of length 52 days, for seven time series composed of the observations for Mondays, Tuesdays, Wednesdays, Thursdays, Fridays, Saturdays and Sundays. Total of seven seasonal ARIMA$(p,d,q)\times(P,D,Q)_{52}$ models where used for Mondays, Tuesdays, Wednesdays, Thursdays, Fridays, Saturdays and Sundays of historical data to obtain out of sample forecasts. The model order and seasonal terms were

determined by autocorrelation and partial autocorrelation functions. Also, significance of model parameters, AIC and log-likelihood values were analysed. This method is called "SARIMA weekday" in this thesis.

For Tuesdays of a strong seasonal and stable time series, SARIMA model coefficient estimates for order and seasonal terms are shown in the Table 7.1. It can be seen that all parameters are significant. Also ACF and PACF graphics in Figures 7.1 and 7.2 show that most of the autocorrelations are removed while there are still small significant values. In Figures 7.1 and 7.2 show lags from 1 to 110. Total periodicity is 52. The functions coded in "R" program in order to apply this method are also explained in the Chapter 8.

Series: tue
ARIMA(2,1,0)(1,1,0)[52]
Call: arima(x = tue, order = c(2, 1, 0), seasonal = list(order = c(1, 1, 0)), xreg = NULL, method = "ML", optim.method = "BFGS", optim.control = list(maxit = 10000), kappa = 1e+06)
Coefficients:

ar1	ar2	sar1
-0.5039	-0.3323	-0.3993
s.e. 0.0537	0.0532	0.0580

sigma^2 estimated as 1910: log likelihood = -1651.98
AIC = 3311.96 AICc = 3312.09 BIC = 3326.99

In-sample error measures:

ME	RMSE	MAE	MPE	MAPE	MASE
-1.1152868	40.4555896	27.9837193	-0.3839014	4.3648808	0.8180948

Figure 7.1. SARIMA model results

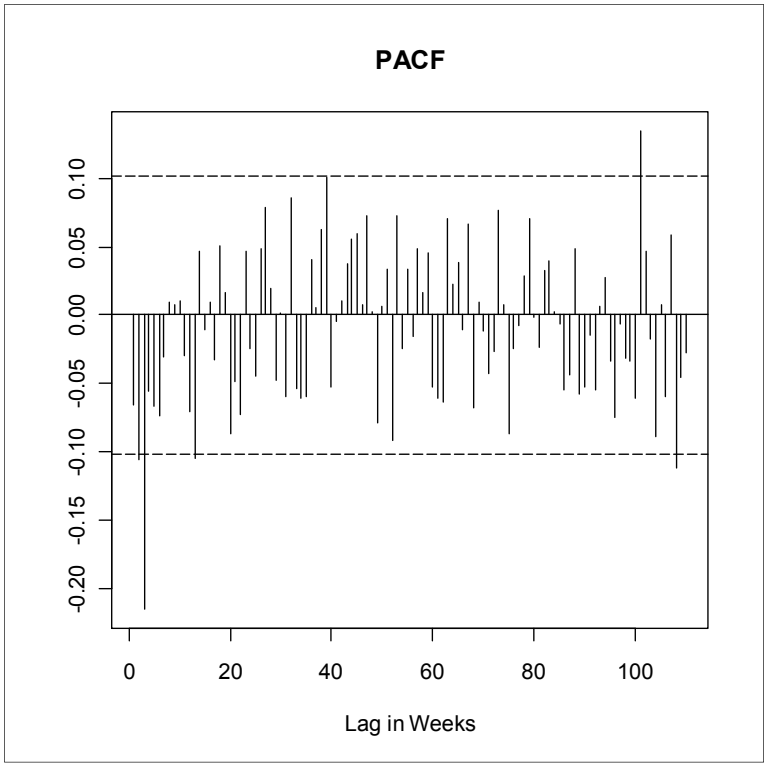

Figure 7.2. Partial ACF for the SARIMA weekday method

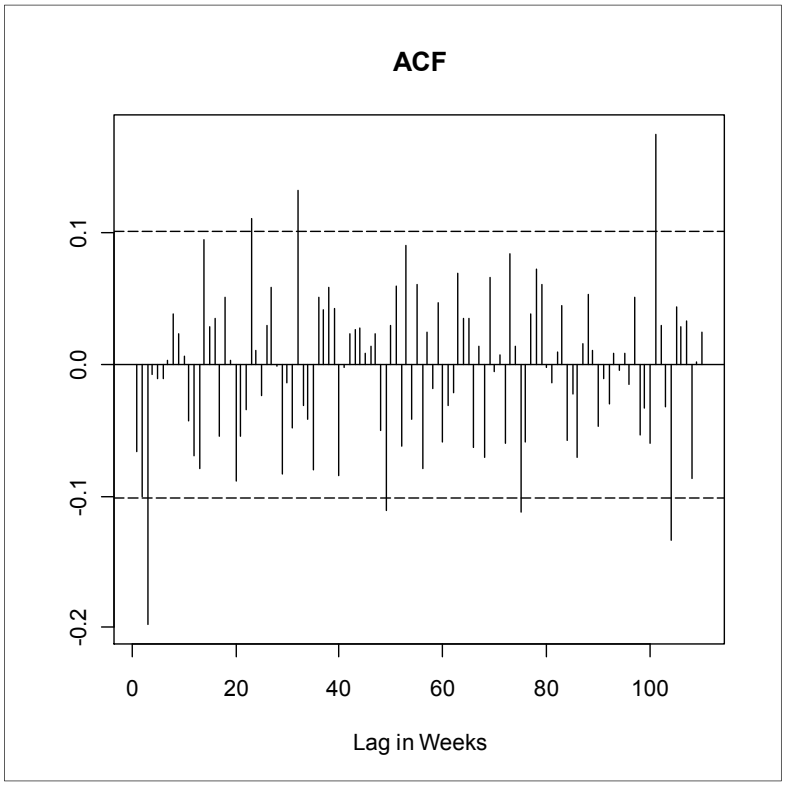

Figure 7.3. ACF for the SARIMA weekday method

8. EXPLANATIONS OF R CODES

In this chapter, the implementations of forecast methods via the forecast functions in "R" programme are explained. The functions are called *naive(), naive_specialday(), naive_weighted_average(), regression(), regspecial_remove_insignif(), regspecial_all(), regspecial_x364()* and *exponential smoothing* functions and *seasonal ARIMA*. The argument set () is a function indication which are given special names.

8.1. Naive Method Functions

8.1.1. Naive and Naive_specialday Functions

For the *naive( )* function, the program simply uses the observations 364 days earlier, and multiplies with the "change ratio" calculated as the sum of the last a few days data in the history data set, divided by the sum of the observations in the same period of the previous year. The "change ratio" is called "factor" in the function. This method does not use any optimization. The length *k* of these windows is 14 days for the "flight" and "passenger" data, and 10 days for the "bank" data. The reason is that, the last two weeks' average value this year versus last year is thought to reflect the change in the values compared to the last year. A window length of 10 days is used for bank data since banks are closed at weekends and the weekends are thus not included in the "bank" time series.

The function *naive( )* uses the arguments "dv, n.ahead, n.window, firstdayofdata". Argument "dv" refers to the time series which is going to be forecasted, "n.ahead" specifies forecasts up to how many days ahead are going to be produced, "n.window" is the data length taken from the last part of the data in order to calculate the "change ratio".

The *naive_specialday( )* function incorporates the special days effects if forecasted days are special days, or if the days 364 days before the forecasted days are special days such as a public holidays. If predicted days are one of the special days, the function takes the value for the same day in the same month of previous year to multiply with the "change

ratio". If forecasted days are not special days but 364 days earlier are special days, then the previous year's special day value is changed with the value for the same day and month in previous year in order to eliminate the special date effect. The function *naive_specialday( )* uses additionally the argument "specialdate".

8.1.2. Naive_weighted_average Function

The *naive_weighted_average( )* function uses not only the observation 364 days before, but also those 357 and 371 days before. It uses *optim( )* function in order to find the optimal factor values. Special days are treated as in the *naive_specialday( )* function. The naive weighted average model assumes that there is a linear and additive relationship between the dependent variable and the lagged dependent variables and the *estimate_generalnaive( )* function is used to determine the coefficients of the lagged dependent variables in the additive equation. In this view, this function can be considered as an additive time-series regression model (on trend-adjusted data) as explained in previous chapters, since model coefficients are found by minimizing sum of squares of forecast errors using optimization.

The *naive_weighted_average( )* is a regression method but it does not use *lm()* function in order to find parameter estimates as we use *lm()* function in other regression models. The difference from other regression models used in this thesis is that this *naive_weighted_average( )* function cuts last part of the data, so as the length of that cut data plus forecast horizon is equal to or less than one year period. Then, for the cut part 364, 357, and 371 days before values are returned as if we are applying naive special day method. But, this time, the coefficients of the three variable are estimated by the *estimate_generalnaive( )* function. In regression models in Chapter 5, the least squares method is used to estimate the model parameters in a model fitting process. In the *naive_weighted_average(),* arguments "coefficient", "dv", "new1", "n.ahead", "n.window", "specialdate", are used. Argument "coefficient" vector indicates regression parameters, "new1" identifies the part of the data that is going to be used in model fitting process. The other parameter are the same with naive special day method. In addition, the other log-linear regression models assume that there is a multiplicative relationship

between regression variables and a linear model was obtained by taking the logarithm of the variables, but in the *naive_weighted_average( )* model we assume that the regression model is linear without taking the logarithm of the variables.

8.2. Regression Functions

8.2.1. Regspecial_remove_insignif, Regspecial_all and Regspecial_x364 Functions

The functions *regspecial_remove_insignif(), regspecial_all()* and *regspecial_x364()* use (0,1) dummy variables for normal versus special days and thus obtain two regression equations one of them for special days when the dummy variable is 1 and the other for normal days when dummy variable is 0. The difference between the *regspecial_x364( )* function and the *regspecial_remove_insignif( )* function is that *the regspecial_x364( )* only uses lagged dependent variable of 364 days before, "wratio", "mathis" and "malast" explanatory variables in order to test other variables' significance and affect to the forecast accuracy. "Mathis" and "malast" vector are used to obtain the "wratio" vector and the "wratio" is the vector of change ratios for every day as explained in Chapter 4.

In *regspecial_remove_insignif ( )* function, the variable which have a coefficient p value larger than 0.1 is eliminated from the regression equation and a new regression equation is used without the insignificant variable. In the *regspecial_all( )* none of the insignificant regression variables is eliminated from the regression equation, so the same variables are used in each simulation period. Regression coefficients found in the regression models are then used to take the exponential of the dependent variables and rows of forecast result matrices are filled with each forecasting process by the inclusion of the new day value.

These functions use *lastyeardate()*, *getlastsamedate()*, *maketsregressiondata()* and *datanahead()* functions in order to return a result matrix of forecasts. The function *lastyeardate( )* returns the same day of the same month of an earlier year. The function *getlastsamedate( )* returns the values of the data one or more years earlier. For example, for the date "2009-05-31", it returns the observation on "2008-05-31". The function

maketsregressiondata( ) returns a data matrix of dependent variables, corresponding special dates as dummy variables, the "wratio" variable (which is the vector of the ratios of the mean value over the last a few days of length "n.window" periods to the mean over same length data 364 days earlier). The function *datanahead( )* returns the regressor variables' future values from one step ahead to "n.ahead" step ahead in order to predict dependent y variables in the log linear regression equation with lagged dependent variables.

8.2.2. Regression Function

The function *regression( )* takes the preferred part of the time series data both from the beginning and from the end part and calculates one-step-ahead to n-step-ahead forecasts "endcut-n.ahead" times. It constructs a forecasts matrix of "n.ahead" columns and "endcut-n.ahead" rows. We mean by the preferred part that, the user of the function can decide to delete the wanted length of data from the beginning of the time series, if the user decides that the time series has more length than he/she needs and if the structure and characteristics in beginning part of the time series have changed by the time due to economic environment or policy changes. This situation is optional and the user of the function may also decide to use the whole time series. Also, if the user of the function wants to do historic forecasting, the wanted length can be cut from the end part of the time series in order to observe the performance of the function.

The function has the argument list "data, n.window, begincut ,n.ahead and endcut". Argument "data" refers to the time series which is going to be forecasted, "n.ahead" explains up to how many days ahead forecasts are going to be produced, "begincut" provides cutting the beginning part of data that is not going to be used and "endcut" is the sum of the number of historic forecast days and n.ahead value. The argument "n.window" is the length of the data window preceding the days forecast is made, which is employed to calculate the change in the mean window data to the mean of a window 364 days earlier.

The function uses a log linear autoregressive regression model with lagged dependent variables in order to predict future values. Autoregressive variables in the model

are 364 days, 357 days, 371 days and 728 days lagged dependent variables since these lagged variables are decided to have relationship with the dependent variable. This regression function also assumes that there is a multiplicative relationship between dependent variable, ratio and lagged dependent variables. Transformation of the data by taking the logarithm of them leads to a log linear regression model to predict future values.

8.3. Exponential Smoothing Functions

In R, there are 2 types of *HoltWinters( )* functions, one of them is additive seasonal and additive trend exponential smoothing function, *singleasat( )*, and the other is multiplicative seasonal and additive trend exponential smoothing function *singlemsat( )*. For the other 26 types of exponential smoothing methods, functions were coded having the same name with the abbreviations of the methods: *doubleasat(), doubleasatauto(), doubleasdmt(), doubleasdmtauto(), doubleasmtauto(), doublemsat(), doublemsatauto(), doublemsdmt(), doublemsdmtauto(), doublemsmtauto(), holtsexposmooth(), multipleasat(), multipleasatauto(), multipleasdmt(), multipleasdmtauto(), multipleasmtauto(), multiplemsat(), multiplemsatauto(), multiplemsdmt(), multiplemsdmtauto(), multiplemsmtauto(), simpleexposmooth(), singleasdmt(), singleasmt(), singlemsdmt(), singlemsmt().*

Apart from the R functions *HoltWinters( )*, the other exponential smoothing functions use the first 728 days (nearly 2 years) of data for setting initialization parameters and the subsequent 730 days (nearly 2 years) of one step ahead forecasts for determining optimal parameters. During the 730 day of optimization period, both 7-day seasonal term and 364-day seasonal term smoothing parameters are optimized in order to give minimum MSE values of one-step-ahead forecasts. All of the exponential smoothing functions could be optimized not only for one-step-ahead forecasts, but also for any given fixed m-step-ahead forecasts, for m =1,2,...,n.

The first part of the function names “single”, “double” and “multiple” implies the seasonality type of the functions. A single seasonal exponential smoothing function has

single seasonal pattern. The cycle length of a single yearly seasonal cycle is 364 days and this seasonality considers the data on the same weekday one year earlier.

In our function names the subsequent letters “as” and “ms” coming after the expressions of “single”, “double” and “multiple” give information whether the seasonality of an exponential smoothing function is additive or multiplicative. The term “as” indicates that seasonality is additive and the term “ms” indicates that seasonality is multiplicative. The terms “at”, “mt”, and “dmt” written after the seasonality terms gives information whether the exponential smoothing function has an additive trend “at”, multiplicative trend “mt” or a damped multiplicative trend “dmt”. The term “auto” used at the end of some function names indicates the autocorrelation correction is used for the forecasting function. For the m step ahead forecast this autocorrelation parameter’s m^{th} power is multiplied with the difference of the last fitted exponential value and the observation value and then, added to the m step ahead forecast result. So, the m^{th} step ahead forecast result is influenced by the one step ahead forecast error for the last observation value.

The parameter of the autocorrelation term is also optimized by the *optim( )* function by minimizing the mean square errors of the one-step-ahead forecasts like for the other seasonal, trend and level parameters. The exponential smoothing function argument “lambda” denotes the parameter vector, “data” denotes the time series used, “n.ahead” denotes the forecast lead time, “initdatayear” denotes the starting period in years for the initialization of parameters. Argument “onestepfnumm” indicates the length of the data that is going to be used in parameter optimization process.

The *estimate_general( )* and *historicsimulation( )* functions provide updates of the parameters and seasonal indexes in a predetermined frequency, for example, every day or every 15^{th} day, and calculates the forecasts over the horizon beginning from the last added day of the data. So, these functions provide a dynamic change in parameter estimates.

8.4. Sarima_Weekday Functions

In the empirical part, data for particular days of the week (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday) are extracted from the data and stored in seven columns made up of sequential Mondays, Tuesdays, Wednesdays,

Thursdays, Fridays, Saturdays and Sundays assuming that each day of the week represents a weekly observation. The length of the season is chosen to be 52 periods and forecasting one-step-ahead for every weekday essentially provides a one-week-ahead forecast for the respective weekday.

"n.ahead" values in prediction functions denote the number of weekly forecasts to obtain. For every day of the week, forecasts are grouped so as to obtain daily forecasts from 1-step-ahead to n-step-ahead. The forecast values of the data are stored in matrices in order to be able to use them in calculation of MAPE values for 1-to-n step ahead forecasts.

Seasonal and non-seasonal model orders are identified by significances of the terms and *acf()* and *pacf()* of *arima()* functions by loading the *forecast, tseries, quadprog, zoo* packages.

8.5. General R Functions

In order to compare different forecasting methods, functions for estimating parameters, calculating forecasts and calculating forecast errors are needed. For exponential smoothing functions *estimate general ( )* and *historicsimulation( )* and for the *naive_weighted_average()* function, *estimate_generalnaive()* and *historicsimulationnaive()* functions are used. The structures of *estimate_general()* and *estimate_generalnaive()* are similar. Also, the structures of *historicsimulation()* and *historicsimulationnaive()* are similar. For the *naive( )* and *naive specialday( )* methods, the *naivesimulation()* function is used. The reason for using a separate historic simulation function for the mentioned naive methods is that the structure of the models are different from the *naive_weighted_average(),* and they do not use parameters. The difference between the *naivesimulation( )* function and the *historicsimulationnaive( )* function is that the former does not use parameter optimization.

The role of the *estimate_general( )* and *estimate_generalnaive( )* functions are to determine optimal parameter values. The *historicsimulation()* functions generally prepare the part of the data that is going to be used according to "historical forecast number", "one

step ahead forecast number", "initialization period" and "forecast lead time", and it cuts off excess data from the beginning of time series data. "Historic forecast number" indicates how many times a new day is going to be added and how many times forecasting is going to be done. This is given a name of "simdays" in *historicsimulation( )* function. "One step ahead forecast number" is the period used for optimization of parameters after the initialization period of seasonal indices for two years. This parameter optimization period is taken as 730 days. This is given a name of "onestepfnum" in *historicsimulation( )* function. "Initialization period" is the period used to obtain seasonal indices. This is given a name of "initdatayear" in functions. "Forecast lead time" indicates how many steps ahead we are going to forecast. This is given a name of "n.ahead" in the functions. It also uses the *estimate_general( )* function in order to find parameters for the forecasting function and prepare a matrix of which row number indicates "historic forecast number" and of which column number indicates "forecast lead time". Historic forecasts are obtained for sequential days. After obtaining forecast result matrices using the functions mentioned above, functions calculating MAPE and MSE are applied.

9. EXPERIMENTS FOR DAILY FORECASTING METHODS

9.1. Daily Time Series Studied

In this thesis, many daily time-series forecasting methods are applied to three different data sets. These three data sets are: 1) daily flight numbers including arrivals and departures in three different regions of Austria, 2) daily passenger numbers in Antalya Airport, and 3) old daily cash withdrawal and deposit amounts of two Austrian branches. Forecast model performances are evaluated and compared by MAPE and MSE for every forecast lead times from 1 to 98 days for flight and passenger data and from 1 to 5 days for the bank data.

The reason for choosing 98 days of forecast lead time for flight and passenger data is that it is important to forecast forthcoming season's customer numbers for enterprises whose workload mainly differs with seasons. So, in order to schedule operational works (such as scheduling labour force) for airports and hotels, it is important to have forecasts up to 3 months ahead. A fixed number of 98 representing 14 weeks is determined to use in order to be able compare all methods with the same forecasts.

For the bank data, it is enough to forecast the next week's cash withdrawal and deposit amount. Banks are closed at the weekends, so forecasts are performed for weekdays and a forecast lead time of 5 days is applied. In this situation, the weekends are extracted from the time series and week lengths are accepted as 5 days and continuity is obtained among weeks. So, instead of using a yearly periodicity of 364 days, 260 days (52 weeks) of yearly periodicity and 5 days of weekly periodicity is used in all functions.

There are three daily time series called ACC, TMA and INN representing daily flight numbers including both arrivals and departures in three different regions of Austria. The length of time series data are 3439 days. The ratio of maximum data to minimum data is around 5. The daily flight data show strong yearly seasonality and the data are comparatively stable. The flight numbers generally increase in summer months and there is an outstanding decrease at holidays such as Christmas. The weekly pattern can be seen

clearly, together with a perceptible yearly seasonality. In addition, the data in every year show similar characteristics with slight level differences but there is a distinct decrease in the last year due to the economic crisis.

The daily time series of Antalya passenger data is called PASS in our experiments. The length of this time series data is 1914 days. The ratio of max/min observations is 40.19. For passenger data, we also see that there is a clear yearly seasonal pattern with a strong increase in summer months. However, as the level or mean of the data increases, weekly fluctuation and variability also increases. Also, on Christmas and the other foreign holidays and on Turkish national holidays, there is a slight increase in some of the years. The reason of this situation can be explained by the profile of Antalya tourists, which consist of visitors from many different nationalities along with Turkish visitors.

Lastly, the third group of data is composed of old daily cash withdrawal at one Austrian branch and cash deposit amounts at another branch of Austria for a period of 1191 days without extracting the weekends. The max/min ratio is 58.95 for cash withdrawal data and 27.93 for cash deposit data. The daily cash withdrawal data of a branch is called “Bank 1”, and the cash deposit data of the other branch is called “Bank 2” in our experiments. These data also reveal seasonality but there is no clear weekly, monthly or yearly seasonality due to highly stochastic or irregular properties of the data. There is high variability in the data within each year, and there is not a clear trend over the years. In addition, as properties of the banking sector, branches of banks are closed on national holidays so, we replaced these days with the values found by the naive method which simply uses the previous week’s values. Hippert et al. (2005) has also replaced electricity load on public holidays by the load observed on a similar day in the week before. Taylor (2010) suggests a method which takes the mean of corresponding periods of the two adjacent weeks, but as we cannot know the following week’s values in advance, we do not use that method. Plots of six time series are shown in Figures 9.1, 9.2, 9.3, 9.4, 9.5 and 9.6.

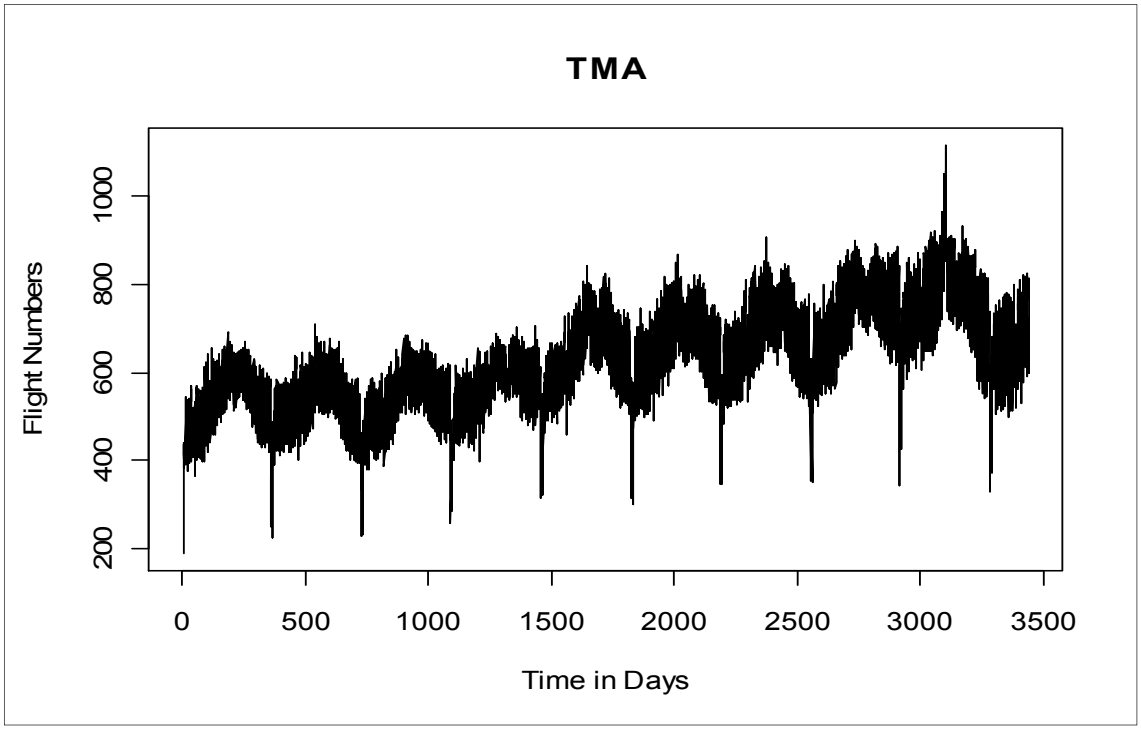

Figure 9.1. Plot of TMA Data

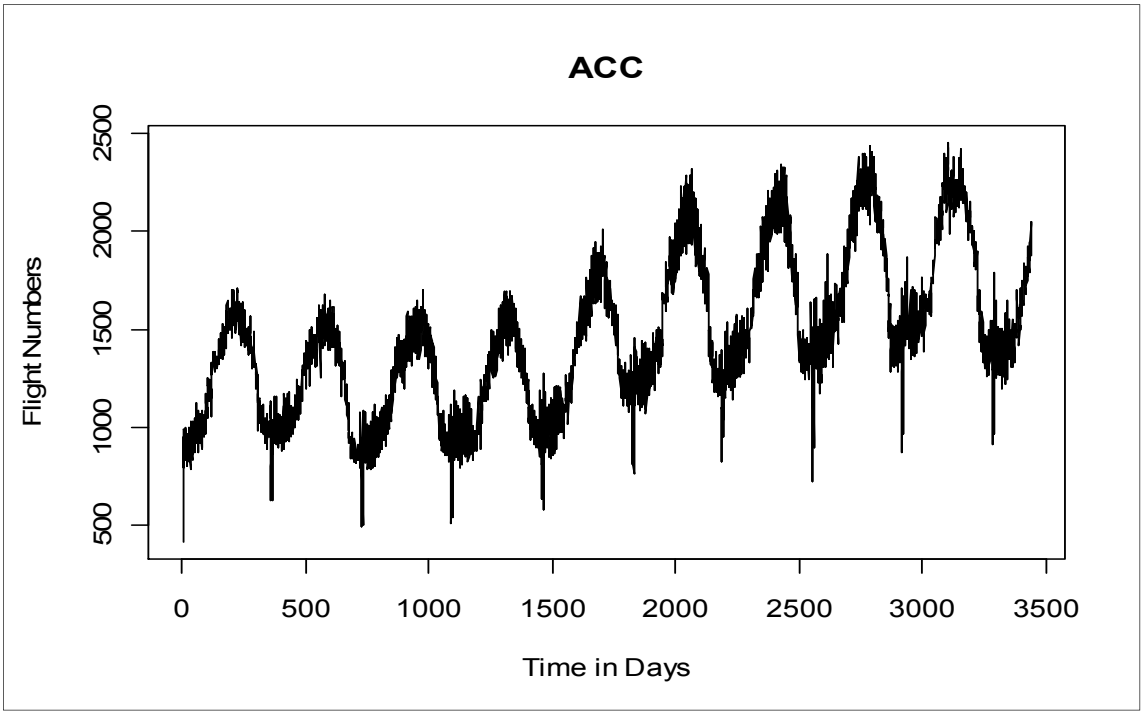

Figure 9.2. Plot of ACC Data

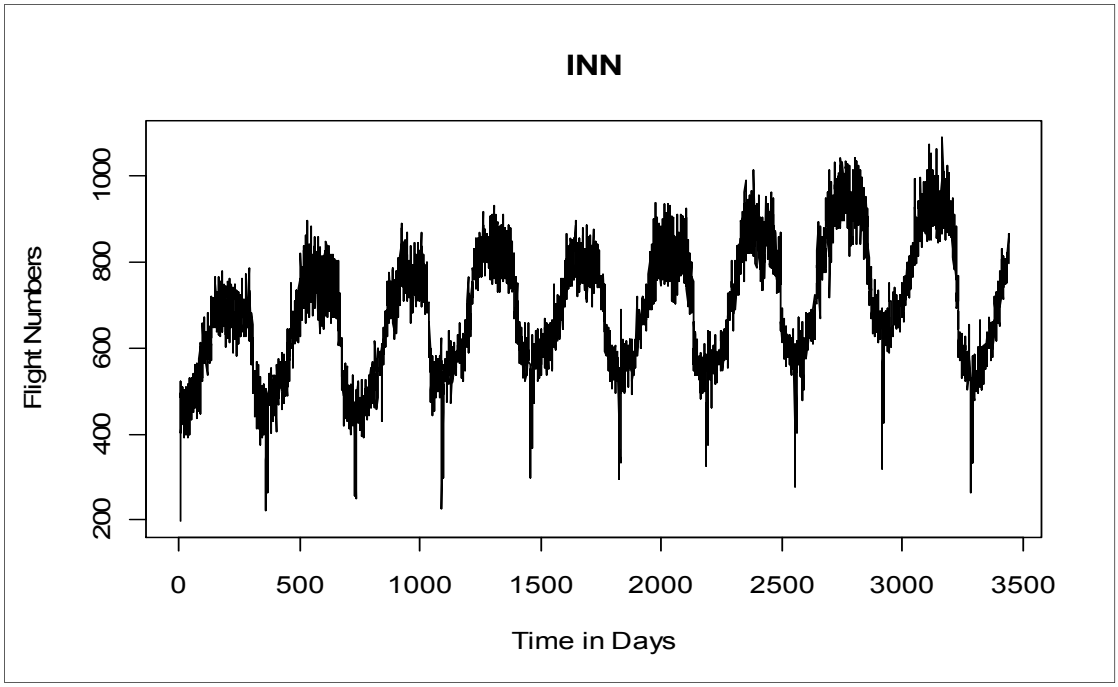

Figure 9.3. Plot of INN Data

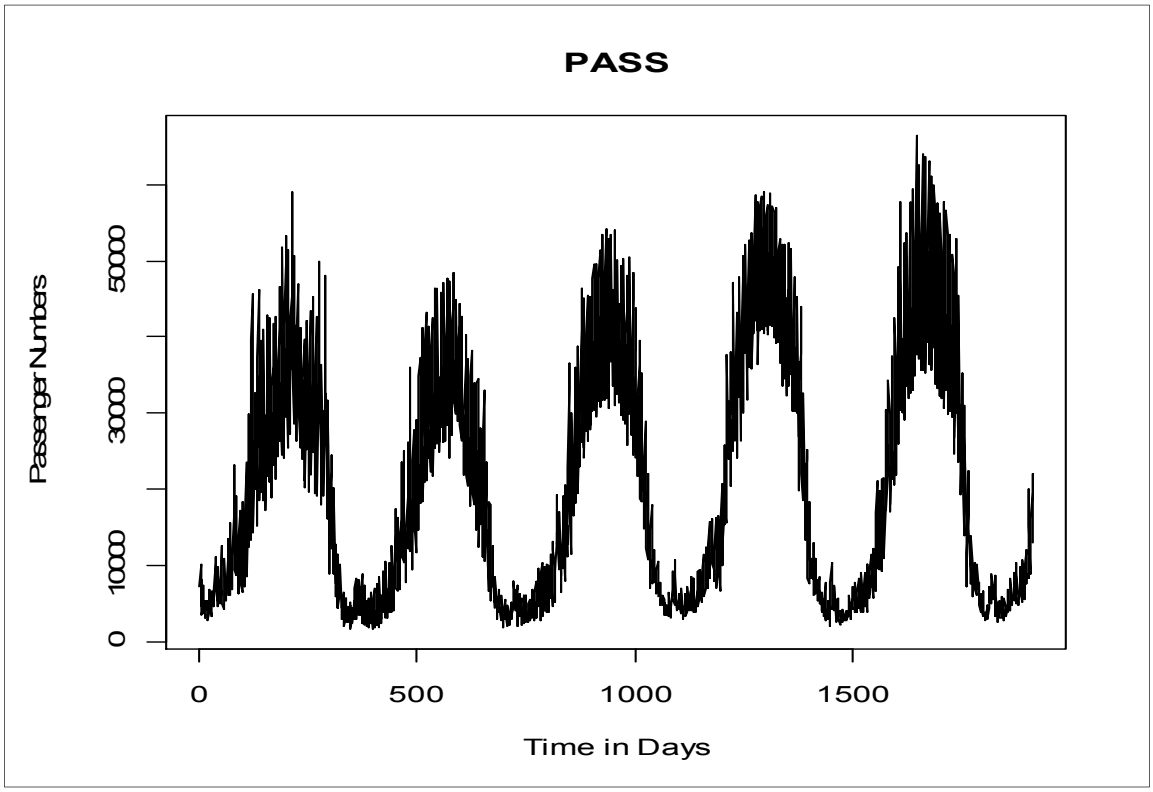

Figure 9.4. Plot of PASS Data

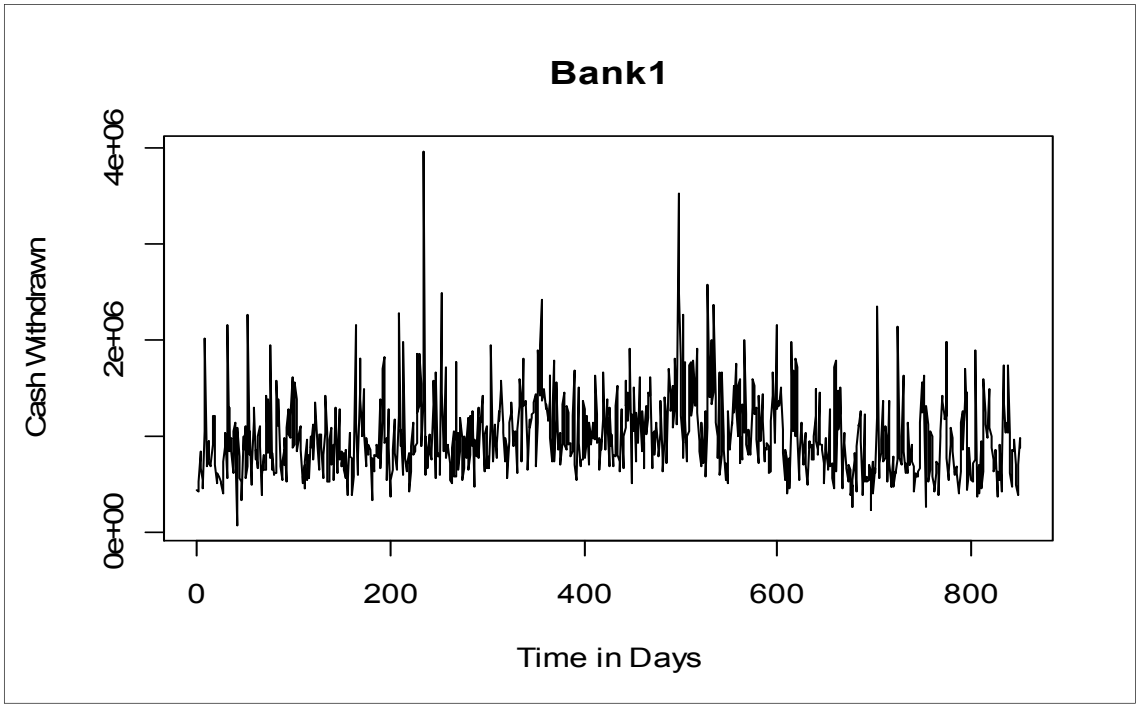

Figure 9.5. Plot of Bank1 Data

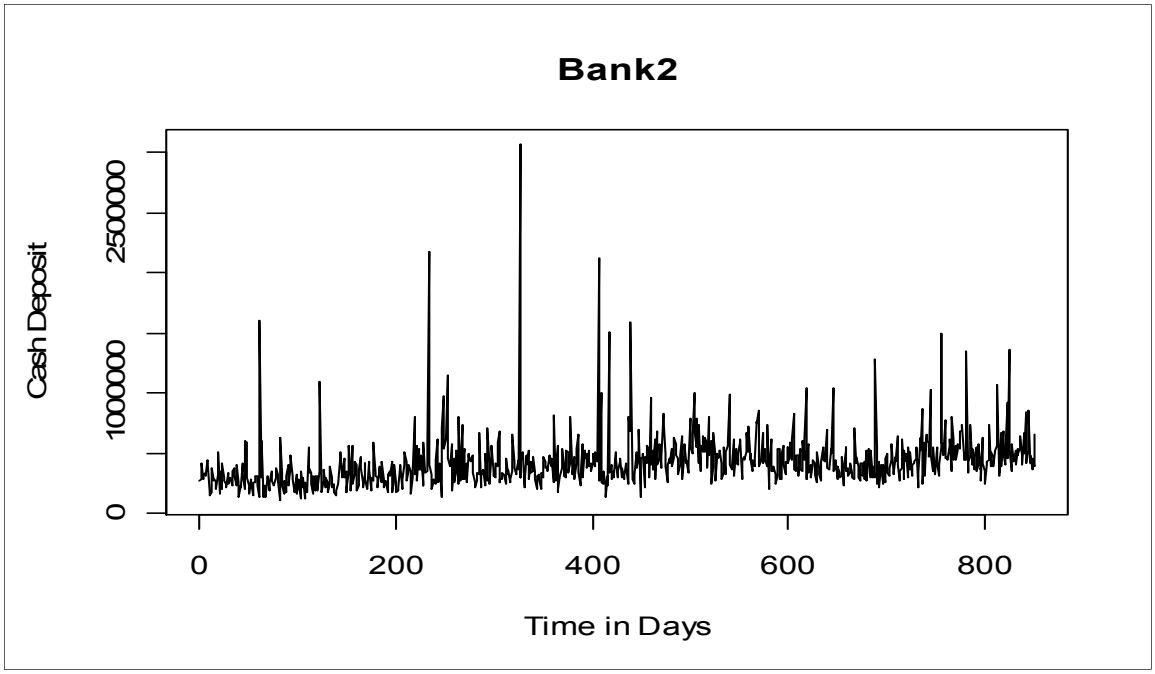

Figure 9.6. Plot of Bank2 Data

9.2. Explanation of Forecasting Methods

9.2.1. Daily Forecast Functions

In our historical forecasting experiments a total of 36 different daily forecasting methods are used; among them, 28 different exponential smoothing methods, 3 different types of naive methods, 4 different regression methods, and 1 SARIMA method are used. The 28 exponential smoothing methods including single, double, multiple, additive and multiplicative seasonality with additive, multiplicative and damped multiplicative trend are coded in R, and two built-in *HoltWinters( )* R functions for single seasonal exponential smoothing are also used.

For ACC data, the seasonal ARIMA(1,1,0)(1,1,0) model is used, for TMA data, the seasonal ARIMA(1,0,1)(1,1,0), for INN data, the seasonal ARIMA(2,1,0)(1,1,0), for Antalya passenger data, the seasonal ARIMA(2,0,0)(1,1,0), for Bank1 cash withdrawal data and Bank 2 cash deposit data, seasonal ARIMA(0,1,1)(1,1,0) models are used for all days of the weeks. The TMA data needed to be re-identified once and seasonal ARIMA(1,0,0)(1,1,0) model is used. The INN data needed to be re-identified with the seasonal ARIMA(1,0,1)(1,1,0) model and the ACC data needed to be re-identified with the seasonal ARIMA(1,0,0)(1,1,0) model. The other PASS, Bank1 and Bank2 did not need to be re-identified.

9.2.2. Explanations of Experiments

The comparison of different methods requires some basic steps and “historic forecasts” of time series data are needed. The historic forecasts are obtained by removing a determined length of data from the end of the data and forecasting these values of the time series. These forecasts are stored in matrices having a row number of simulation days and column number of forecasts steps and these can then be compared with the observed values. Historic forecasting allows for measuring the performance of forecasting models based on previously observed actual values.

The “simulation” term is used in this thesis to imply obtaining recursive forecasts for sequentially included days of a time series for a determined time interval, as if time was passing by. The “historical simulation lengths” indicate how many times an additional day is added to the last day of the time series data used and how many times forecast generation is performed.

The historical simulation length differs for the three different data sets because of the differences in the lengths. For flight data with 3439 observations, we used 365 days, which is less than the final simulation length of 750 since optimization of parameters takes a long time; for passenger data with 1914 observations we used 358 days and for bank data with 850 observations without weekends, we used 60 days for the historic simulation length.

First of all, in historical forecasting experiments, 26 exponential smoothing methods were compared, with optimization of smoothing parameters every 20^{th} day. Parameters were optimized every 20 times historic forecasts were obtained, and at other times, the previously obtained parameters are used in exponential smoothing methods except from *singleasat* and *singlemsat* exponential smoothing methods.

The reason for not optimizing the parameters every day for all exponential smoothing techniques is that the optimization process takes a long time. In this way, when there is a change in the behaviour of the time series, parameters are adjusted according to the new situation not every day but every 20^{th} day.

For the flight and passenger data, maximum forecast lead time is 98 days, and for bank data, maximum forecast lead time is 5 days, because of sector properties as previously mentioned.

Then, for each different forecast lead time, the exponential smoothing method giving the smallest MAPE was chosen in order to compare it with the other mentioned forecasting methods. MSE of the forecast results are also examined.

For the specific exponential smoothing methods chosen for specific forecast lead times (and in fact, also for all other forecasting methods), the historic forecasting results

were then re-obtained, this time with daily optimization of parameters and updating of seasonal indices.

For testing if the exponential smoothing method giving the minimum MAPE using optimization every 20^{th} day is also giving the minimum MAPE with daily optimization, other exponential smoothing methods which also resulted in small MAPE values were chosen and re-run with every day optimization of the parameters.

9.3. Experimental Results

The aim of these forecasting experiments is to compare different forecasting methods for daily seasonal data and to conclude which single forecasting method gives good forecasting results across all forecast horizons and is more convenient for the different data used.

9.3.1. The Results for Flight Data

The flight data results for the different exponential smoothing functions are obtained. The results are the mean of MAPE's for forecast lead times up to 98 days ahead with optimization of parameters every 20^{th} day. My goal (for now, initially) is to find a single forecasting method that gives good forecasting results across all possible forecast horizons. Even though management is interested in specific period ahead forecasting performances, general mean of MAPE's for all forecasting horizons will provide us a general idea about forecasting performances. The three different time series results are given in Table 9.1. In Table 9.1, *singleasat* and *singlemsat* models are not included since these two single additive and multiplicative seasonal exponential smoothing models are the most commonly used exponential smoothing models, and will be compared with all other (non-exponential-smoothing) forecasting methods.

Table 9.1. MAPE for Exponential Smoothing Methods for the Flight Data

Exponential Smoothing Methods	**ACC mean MAPE**	**TMA mean MAPE**	**INN mean MAPE**	**Exponential Smoothing Methods**	**ACC mean MAPE**	**TMA mean MAPE**	**INN mean MAPE**
doubleasat	0.11105	0.08667	0.11850	multipleasdmt	0.06592	0.10489	0.10022
doubleasatauto	0.11121	0.08759	0.11856	multipleasdmtauto	0.06621	0.10497	0.10111
doubleasdmt	0.06676	0.10470	0.09998	multipleasmtauto	0.12214	0.08677	0.12031
doubleasdmtauto	0.06777	0.10556	0.10015	multiplemsat	0.11601	0.08800	0.10924
doubleasmtauto	0.12196	0.08735	0.12027	multiplemsatauto	0.11622	0.08819	0.11004
doublemsat	0.11678	0.08755	0.11004	multiplemsdmt	0.06686	0.09978	0.09744
doublemsatauto	0.11724	0.08812	0.10874	multiplemsdmtauto	0.06764	0.09975	0.09857
doublemsdmt	0.06757	0.09892	0.09667	multiplemsmtauto	0.12761	0.08822	0.11076
doublemsdmtauto	0.06759	0.10005	0.09670	simpleexposmooth	0.20360	0.14854	0.22426
doublemsmtauto	0.12833	0.08813	0.11004	singleasdmt	0.05379	0.08822	0.09686
holtsexposmooth	0.20133	0.13060	0.25922	singleasmt	0.12152	0.07211	0.11696
multipleasat	0.11083	0.08666	0.11904	singlemsdmt	**0.05265**	0.08795	**0.09459**
multipleasatauto	0.11110	0.08700	0.11929	singlemsmt	0.12620	**0.06364**	0.10634

For ACC flight data, *doubleasdmt, doubleasdmtauto, doublemsdmt, doublemsdmtauto, multipleasdmt, multipleasdmtauto, multiplemsdmt, multiplemsdmtauto, singleasdmt, singlemsdmt* models give better results. *Singlemsdmt* model gives the smallest MAPE on the average for forecast lead times. There is no significant difference between multiplicative seasonal and additive seasonal models since there is not increasing amplitude in seasonal variations of ACC data. It can be observed that damped multiplicative trend exponential smoothing methods generally give smaller MAPE whereas double and multiple seasonal exponential smoothing methods and methods with autocorrelation terms do not outperform single seasonal and damped multiplicative trend methods.

The lowest-MAPE *singlemsdmt* forecasts (and the commonly used *singleasat* and *singlemsat* forecasts) are compared next with all the other forecasting methods, this time for various forecast horizon windows.

Table 9.2. MAPE for Forecasting Methods for the ACC Data

Methods ACC	**mean MAPE 1-4 Weeks**	**mean MAPE 5-8 Weeks**	**mean MAPE 9-14 Weeks**	**overall mean**
Singleasat	0.03890	0.04471	0.04887	0.04483
Singlemsat	0.04006	0.04683	0.05195	0.04709
Singlemsdmt	0.04458	0.04661	**0.04795**	0.04661
Naive	0.04579	0.05018	0.05641	0.05160
Naive_specialday	0.04238	0.04656	0.05283	0.04806
Naive_weighted_average	0.05427	0.05805	0.06528	0.06007
Regspecial_remove_insignif	0.05180	0.05744	0.06446	0.05884
Regspecial_all	**0.03716**	**0.04172**	0.05049	**0.04418**
Regspecial_x364	0.04024	0.05071	0.06574	0.05416
Regression	0.04131	0.04590	0.05266	0.04749
Sarima_weekday	0.04719	0.05263	0.05705	0.05297

Among other forecasting methods, it can be seen that the *singleasat* (single additive seasonal and additive trend exponential smoothing) and *regspecial_all* methods outperform other forecast methods, and *regspecial_all* method gives the smallest MAPE value for up to 8 weeks ahead forecast and for the overall mean. Removing insignificant variables from regression equation does not increase the model's forecast performance. Since, there is an observable special day effect *regspecial_all* model has a better forecast performance. For longer forecast horizons *singlemsdmt* model obtains better results. The reason is that the dampening parameter adjusts growth rate by impeding exponential increases over time by taking dampening parameter values between 0 and 1 if the growth rate is greater than 1 for the last day and by impeding exponential decreases if the growth rate is less than 1 for the last day.

The plot in Figure 9.7 of the MAPEs for specific forecast horizons of the forecasting techniques noted in Table 9.2 indicates that the smallest forecast errors are for the *regspecial_all* model, for all forecast horizons up to about 60 days ahead. For longer forecast horizons, *singlemsdmt* model provides the most accurate forecasts, with *singleasat*

a close second. It can be observed that MSE plot of ACC in Figure 9.8 corresponds to MAPE plot of ACC.

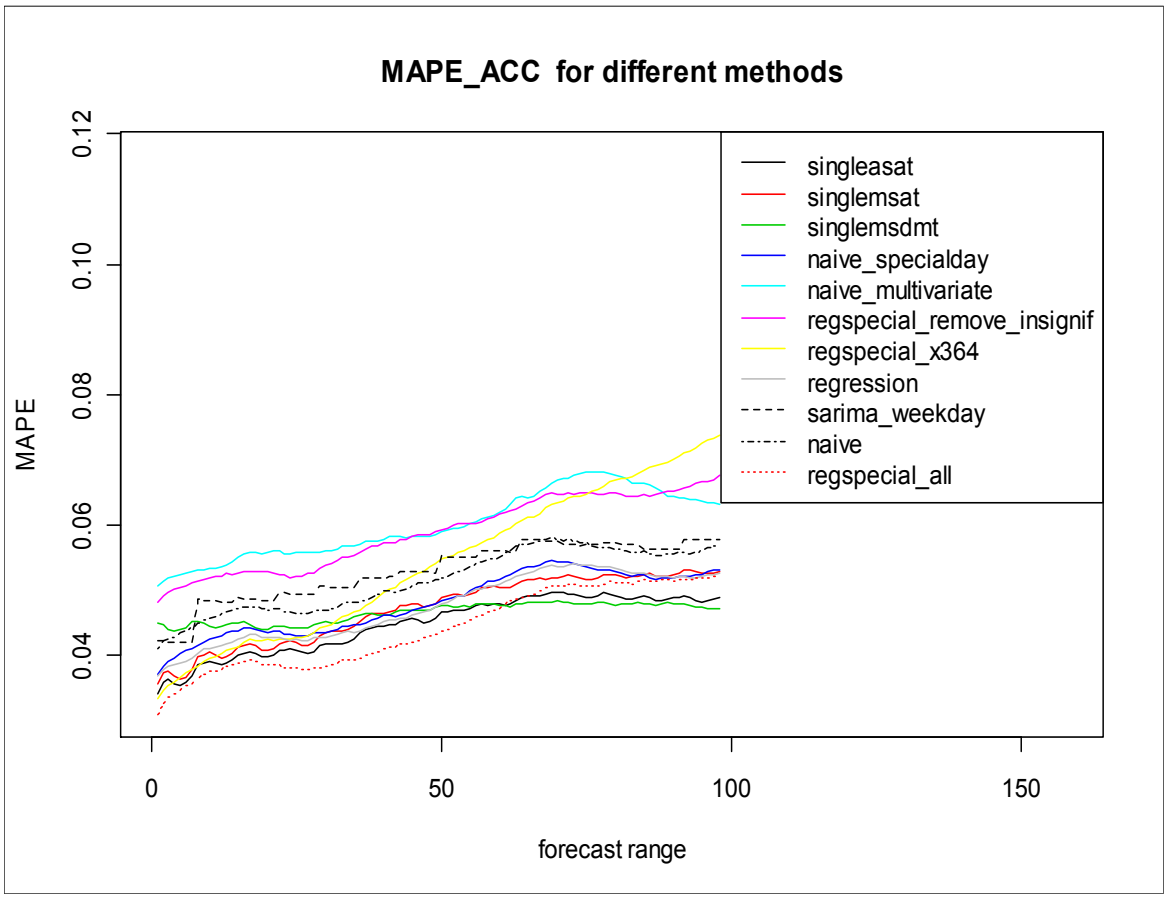

Figure 9.7. MAPE plot for the Forecasting Methods for the ACC Data

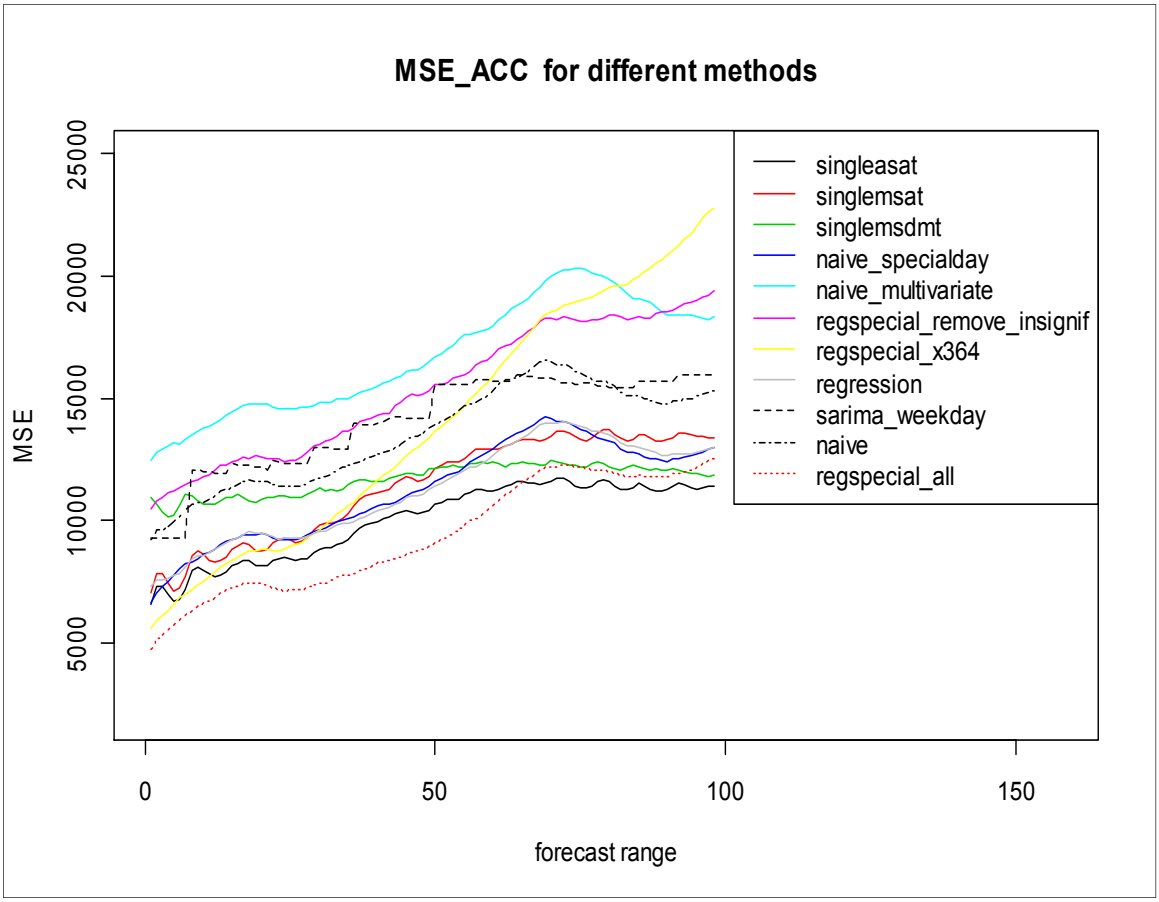

Figure 9.8. MSE plot for the Forecasting Methods for the ACC Data

For TMA flight data, *singlemsmt* and *singleasmt* models give considerably lower MAPE results among the exponential smoothing methods reported in Table 9.1. *Singlemsmt* model gives the smallest average MAPE over all forecast lead times. However, it can be seen in Table 9.3 that *singlemsmt* model does not outperform the *singleasat* and *singlemsat* exponential smoothing models. On the other hand, exponential smoothing, naive and *sarima_weekday* models are outperformed by *regspecial_all* and *regspecial_x364* models. In general, *singleasat*, *simglemsat*, *naive, naive_specialday* and regression models' forecasts show a good performance.

For TMA time series, seasonal variance does not increase dramatically when the mean increases; and when we remove seasonal effects, the data are almost stable. For these reasons, it is reasonable that the additive trend exponential smoothing and *naive_specialday* methods also give small forecast errors.

The *sarima_weekday* does not give satisfactory results. The seasonal ARIMA model orders are required to be re-identified during simulation period. From this situation, it can be understood that ARIMA model requires to be revised to fit to the time series and the model is not fully capable to explain the stochastic components. MSE plot and MAPE plots for TMA data seem to be corresponding with each other.

Table 9.3. MAPE for Forecasting Methods for the TMA Data

Methods TMA	**mean MAPE 1-4 Weeks**	**mean MAPE 5-8 Weeks**	**mean MAPE 9-14 Weeks**	**overall mean**
Singleasat	0.03972	0.04501	0.05177	0.04639
Singlemsat	0.04027	0.04579	0.05378	0.04764
Singlemsmt	0.04608	0.05071	0.05798	0.05250
Naive	0.04558	0.05251	0.06108	0.05420
Naive_specialday	0.04036	0.04707	0.05576	0.04888
Naive_weighted_average	0.05195	0.05318	0.05790	0.05485
Regspecial_remove_insignif	0.05585	0.06050	0.06723	0.06206
Regspecial_all	**0.03499**	**0.04134**	**0.04961**	**0.04307**
Regspecial_x364	0.03846	0.04537	0.05322	0.04676
Regression	0.04237	0.04828	0.05595	0.04988
Sarima_weekday	0.05533	0.07199	0.08114	0.07115

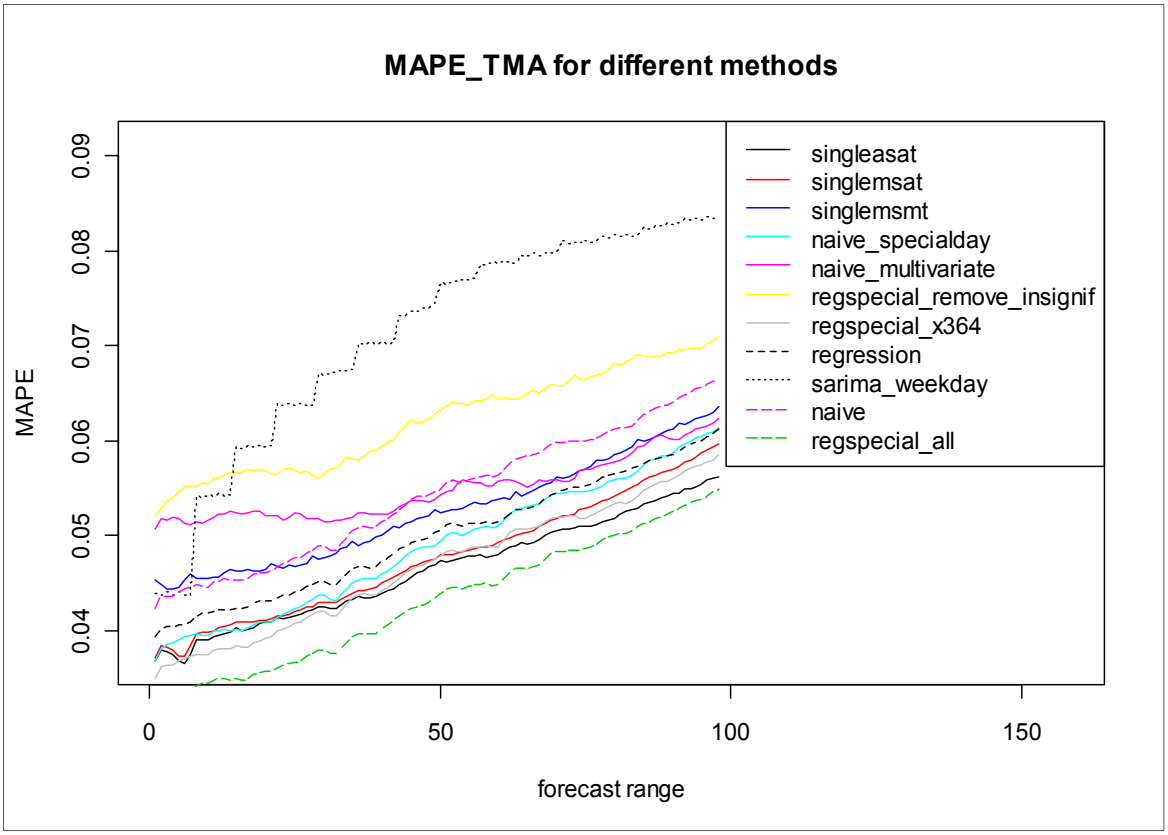

Figure 9.9. MAPE plot for the Forecasting Methods for the TMA Data

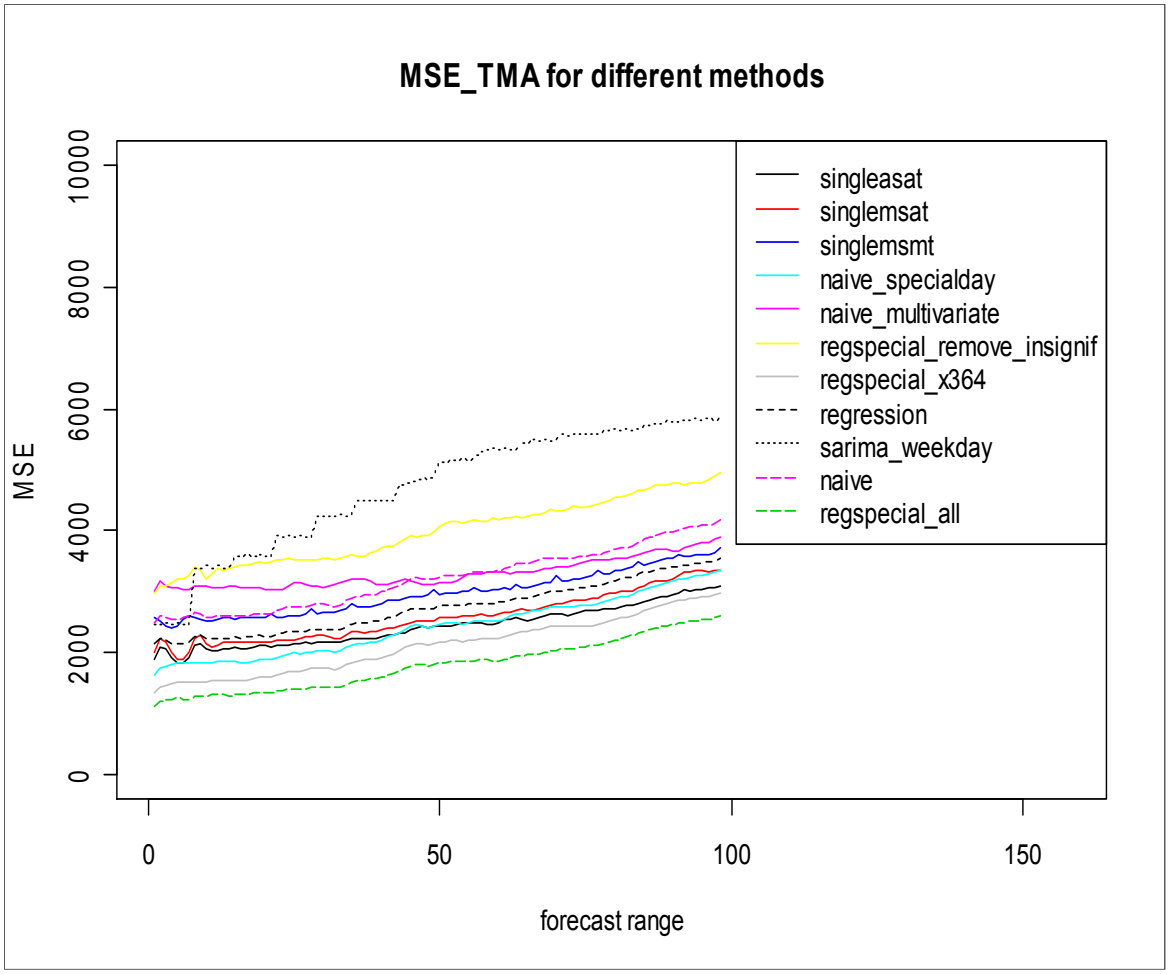

Figure 9.10. MSE plot for the Forecasting Methods for the TMA Data

For INN flight data, Table 9.4 shows that *doublemsdmt, doublemsdmtauto, multiplemsdmt, multiplemsdmtauto, singleasdmt* and *singlemsdmt* models give small forecast errors, and the method giving the smallest MAPE is *singlemsdmt*. As for ACC data, *regspecial_all* is the best forecast method for up to 8 weeks ahead forecasts and *singlemsdmt* model is the best method for longer lead times. The *naive_weighted_average* method gives the largest errors, since there is a strong yearly seasonality in time series data. In this time series, in general single seasonal and additive trend exponential smoothing and regression methods give better forecast results. The *naive* method does not give as small forecast errors in INN data as in TMA data, since weekly seasonal indices variation is higher than for TMA data. The weekly seasonal indices variances are 1.078 and 1.086 for ACC and INN data and 1.054 for TMA data. MSE and MAPE plots for INN data seem be corresponding with each other.

Table 9.4. MAPE for Forecasting Methods for the INN Data

Methods INN	**Mean MAPE 1-4 Weeks**	**Mean MAPE 5-8 Weeks**	**Mean MAPE 9-14 Weeks**	**overall mean**
Singleasat	0.04804	0.05828	0.06856	0.05976
Singlemsat	0.04904	0.05905	0.06838	0.06019
Singlemsdmt	0.05398	0.06190	**0.06763**	0.06209
Naive	0.05591	0.06646	0.07778	0.06830
Naive_specialday	0.05063	0.06131	0.07371	0.06358
Naive_weighted_average	0.07235	0.08316	0.08880	0.08249
Regspecial_remove_insignif	0.05203	0.06258	0.07490	0.06484
Regspecial_all	**0.04368**	**0.05593**	0.07044	**0.05865**
Regspecial_x364	0.04950	0.07103	0.09636	0.07573
Regression	0.04935	0.05892	0.06935	0.06066
Sarima_weekday	0.05735	0.06786	0.07639	0.06852

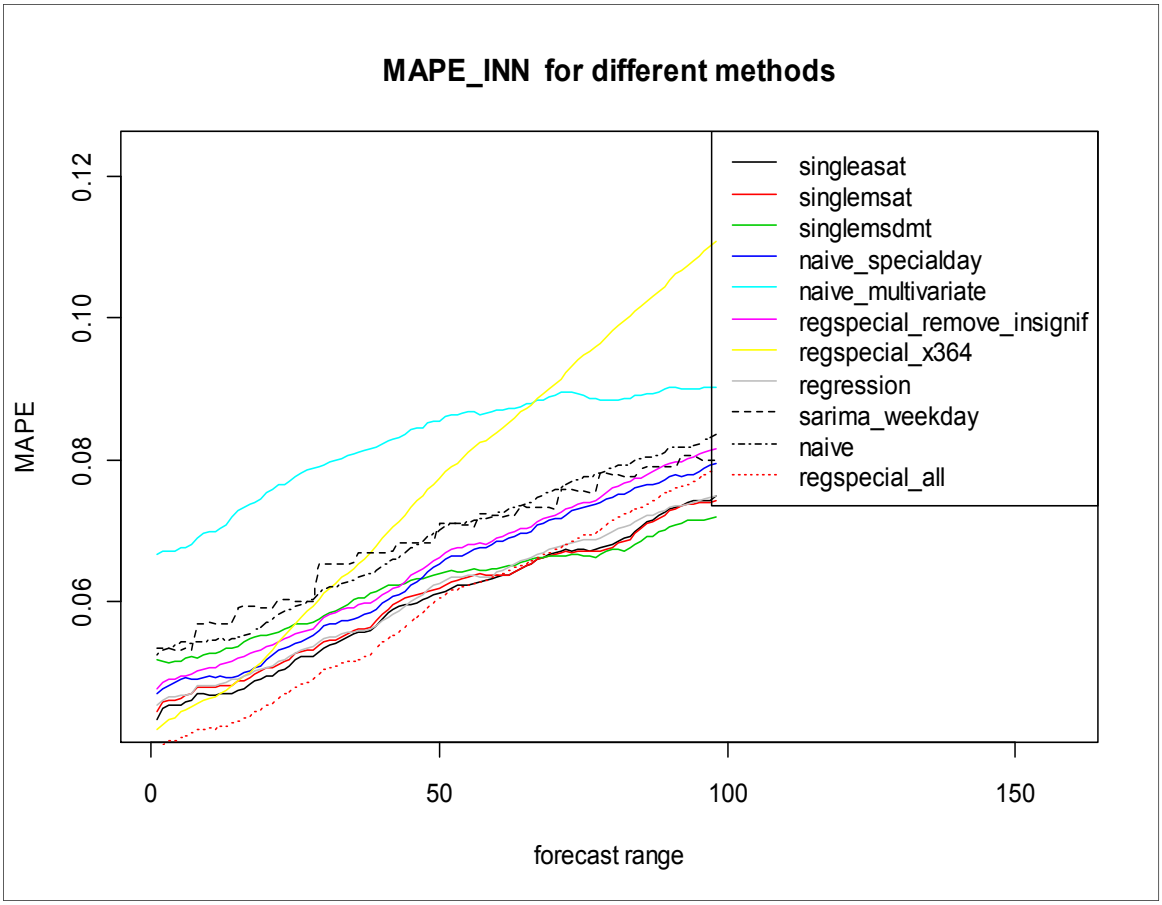

Figure 9.11. MAPE plot for the Forecasting Methods for the INN Data

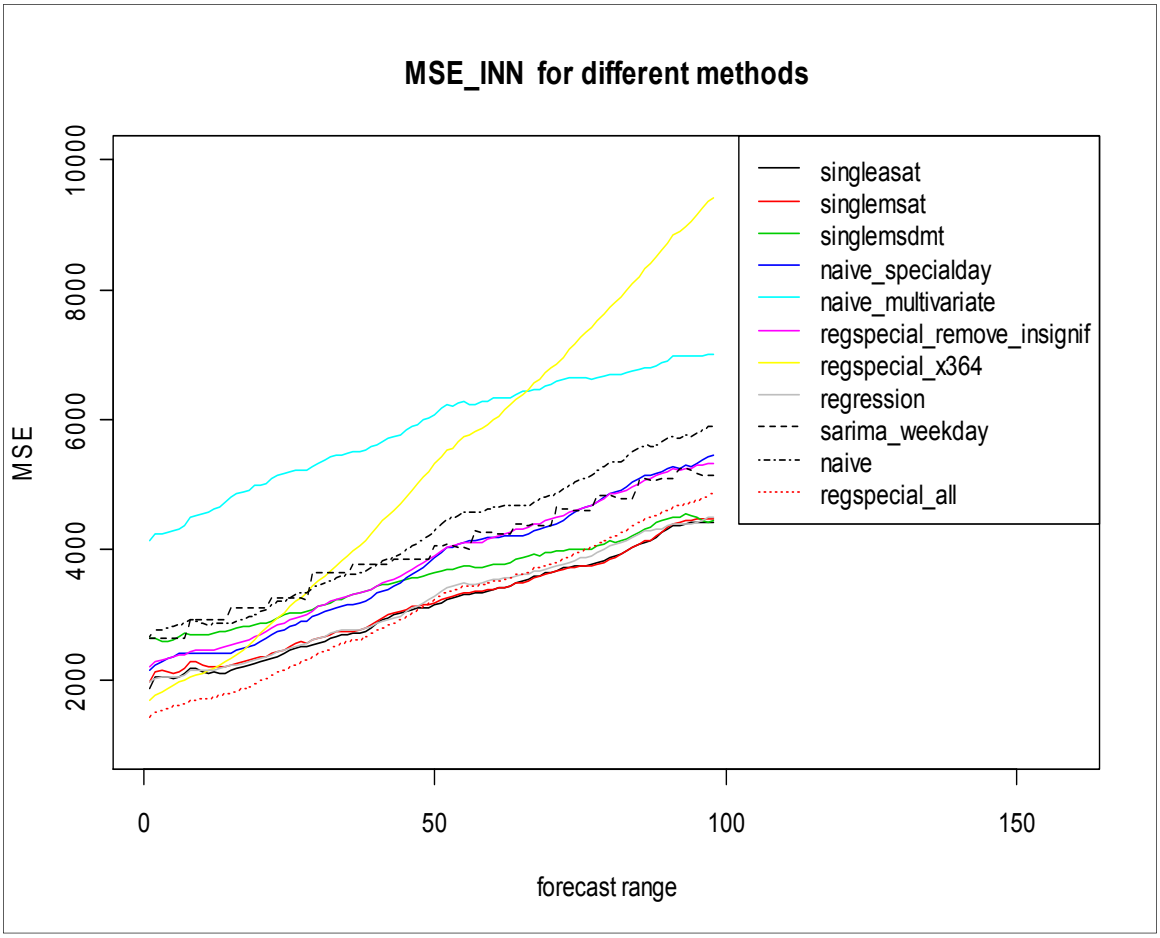

Figure 9.12. MSE plot for the Forecasting Methods for the INN Data

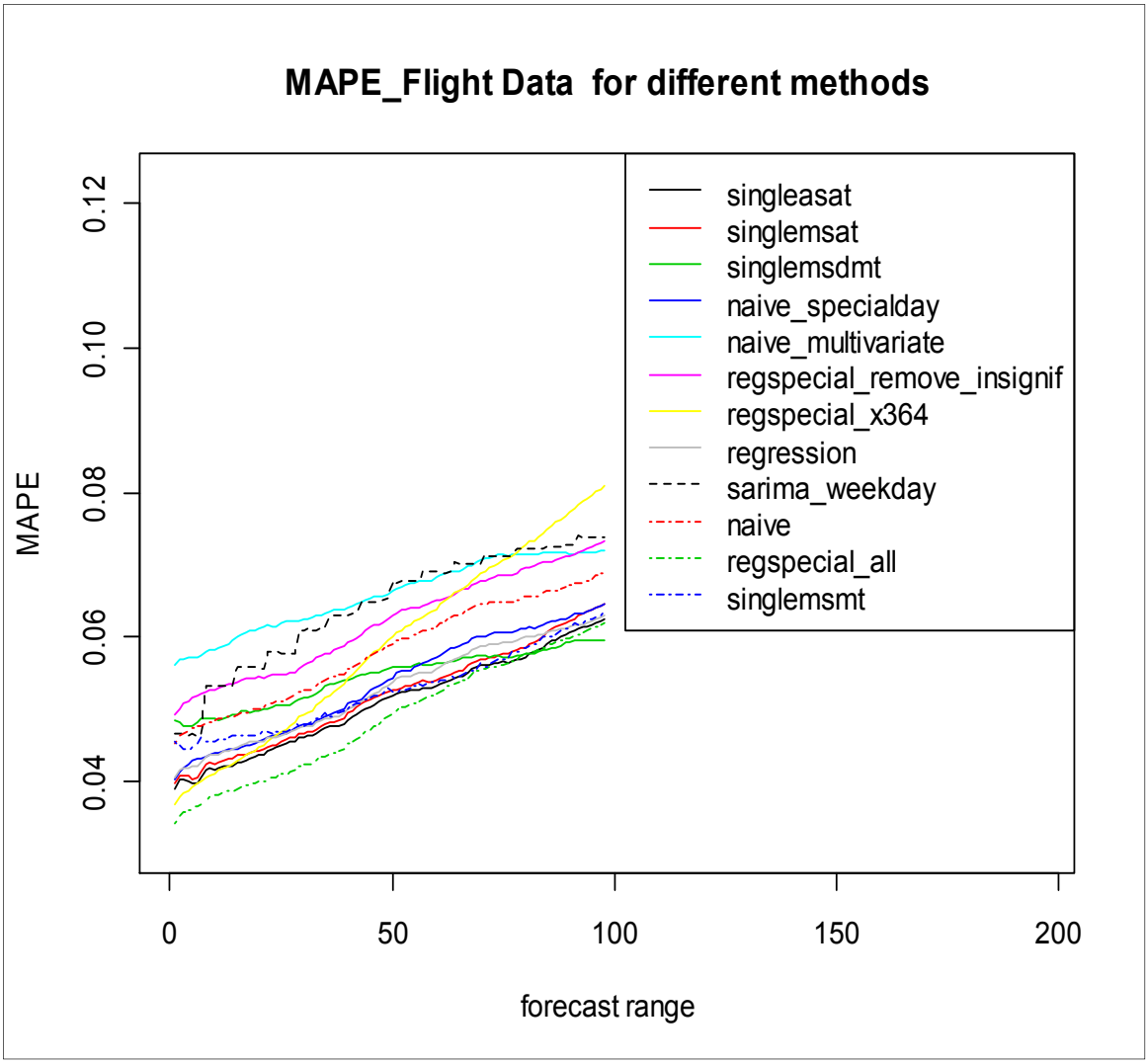

Figure 9.13. MAPE plot for the Forecasting Methods for the Flight Data

Overall, *naive, naive_specialday, regression, regspecial_all, singleasat* and *singlemsat* models produce good forecasts for the flight data. Also, the models in which there is a special day adjustment outperform the models which do not include special day adjustments. After removing the seasonal component from the flight data, the remaining data are quite stable and it is reasonable that additive trend exponential smoothing models and naive models are successful.

9.3.2. The Results for Passenger Data

The passenger data results for exponential smoothing methods are given in Table 9.5. It can be observed that *doublemsdmt, multiplemsdmt* and *singlemsdmt* models outperform other exponential smoothing methods. *Singlemsdmt* model gives the smallest MAPE value for the average of all forecast lead times.

Table 9.5. MAPE for Exponential Smoothing Methods for the Passenger Data

Exponential Smoothing Methods	**PASS mean MAPE**	**Exponential Smoothing Methods**	**PASS mean MAPE**
doubleasat	0.90599	multipleasdmt	0.55424
doubleasatauto	0.90890	multipleasdmtauto	0.56009
doubleasdmt	0.54325	multipleasmtauto	0.94368
doubleasdmtauto	0.54930	multiplemsat	0.45216
doubleasmtauto	0.94584	multiplemsatauto	0.45303
doublemsat	0.44945	multiplemsdmt	0.36608
doublemsatauto	0.45251	multiplemsdmtauto	0.37319
doublemsdmt	0.35833	multiplemsmtauto	0.45319
doublemsdmtauto	0.36143	simpleexposmooth	1.33587
doublemsmtauto	0.45202	singleasdmt	0.40377
holtsexposmooth	1.64104	singleasmt	0.90093
multipleasat	0.90710	singlemsdmt	**0.28936**
multipleasatauto	0.91169	singlemsmt	0.41078

Since, *doublemsdmt*, *multiplemsdmt* and *multiplemsat* models give relatively small forecast errors, these three models are also included in the comparison with the other forecasting methods.

For Antalya passenger data, *regression* and *sarima_weekday* methods produce most accurate forecasts. Among the exponential smoothing methods, *singlemsat* method gives the smallest forecast errors, but as compared to *sarima_weekday* and regression methods, it

still performs poorly. Furthermore, the special days' effect is not clear in the time series data. As a result, methods considering special day effects give less desirable results according to methods not doing special days adjustments.

Table 9.6. MAPE for Forecasting Methods for the Passenger Data

Methods PASS	**mean MAPE 1-4 Weeks**	**mean MAPE 5-8 Weeks**	**mean MAPE 9-14 Weeks**	**overall mean**
Singleasat	0.17418	0.26263	0.39342	0.29341
Singlemsat	0.14343	0.15142	0.16053	0.15304
Singlemsdmt	0.20131	0.23013	0.25001	0.23042
Doublemsdmt	0.22829	0.20900	0.18249	0.20315
Multiplemsat	0.21753	0.22727	0.23034	0.22580
Multiplemsdmt	0.20890	0.21751	0.22083	0.21648
Naive	0.15131	0.14084	0.15506	0.14993
Naive_specialday	0.15841	0.14858	0.16181	0.15706
Naive_weighted_average	0.17670	0.15841	0.14887	0.15955
Regspecial_remove_insignif	0.15104	0.13524	0.13772	0.14081
Regspecial_all	0.14485	0.13344	0.14152	0.14016
Regspecial_x364	0.15423	0.14392	0.15828	0.15302
Regression	0.14194	**0.13096**	0.14237	**0.13899**
Sarima_weekday	0.13403	0.14541	**0.13701**	**0.13856**

The Antalya passenger time series shows high autocorrelation between seasonally lagged values; so, using an autoregressive seasonal ARIMA (2,0,0)(1,1,0) model increased forecast performance. The *naive, naive_specialday* and *regspecial_x364* models also give satisfying results because there is a strong autocorrelation between 364-day-lagged values. Also, the regression methods use seasonally lagged values and fits the model by the least squares method.

For this time series, *singleasat* model results are not satisfying because the seasonality is multiplicative in Antalya passenger data. Multiple and double seasonal exponential smoothing methods are not successful, because yearly seasonality is superior to weekly seasonality. MSE and MAPE plots of Antalya passenger data show that, *singleasat* function is not performing poorly in MSE plot as in MAPE plot. The reason for it will be that *singleasat* performs poorly in winter, in which the number of flights fairly decline.

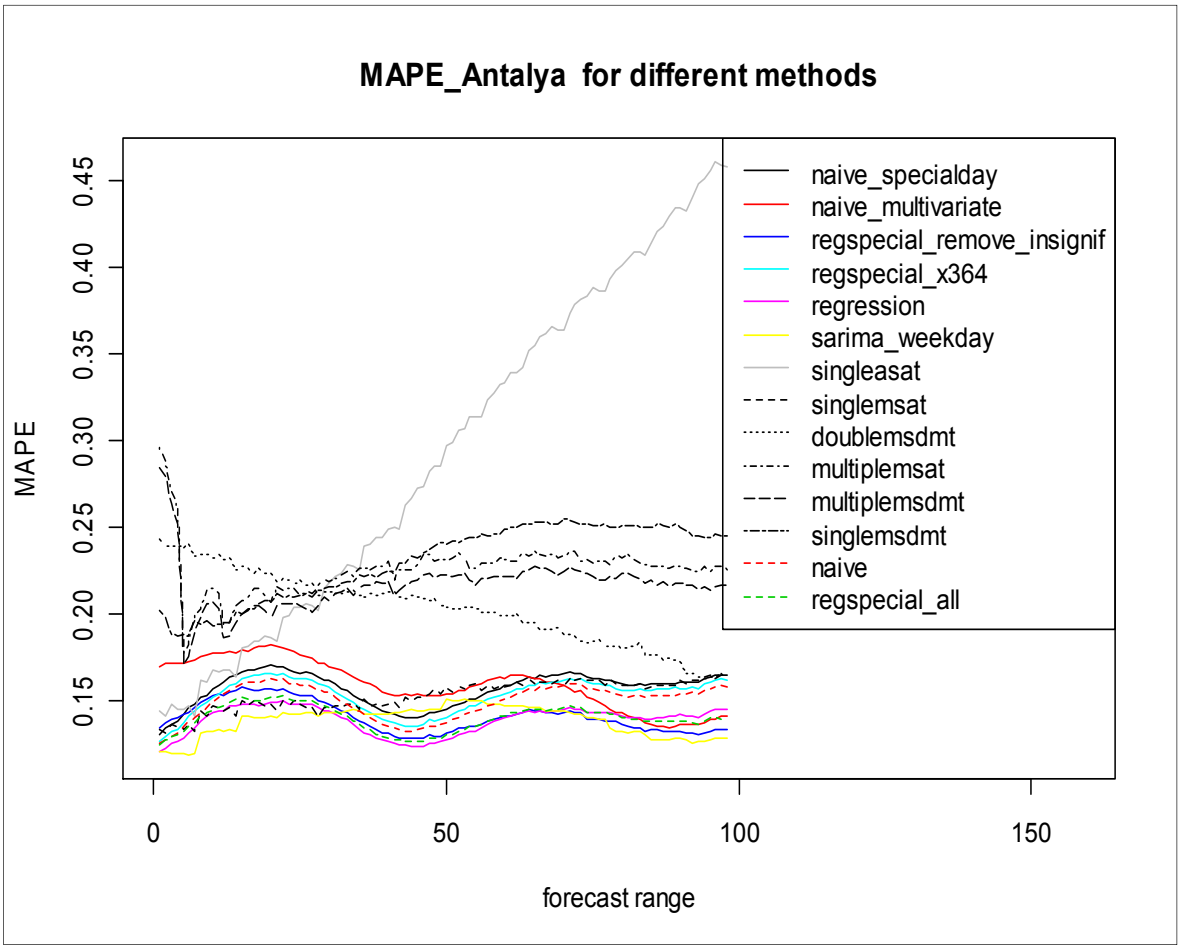

Figure 9.14. MAPE plot for the Forecasting Methods for the Passenger Data

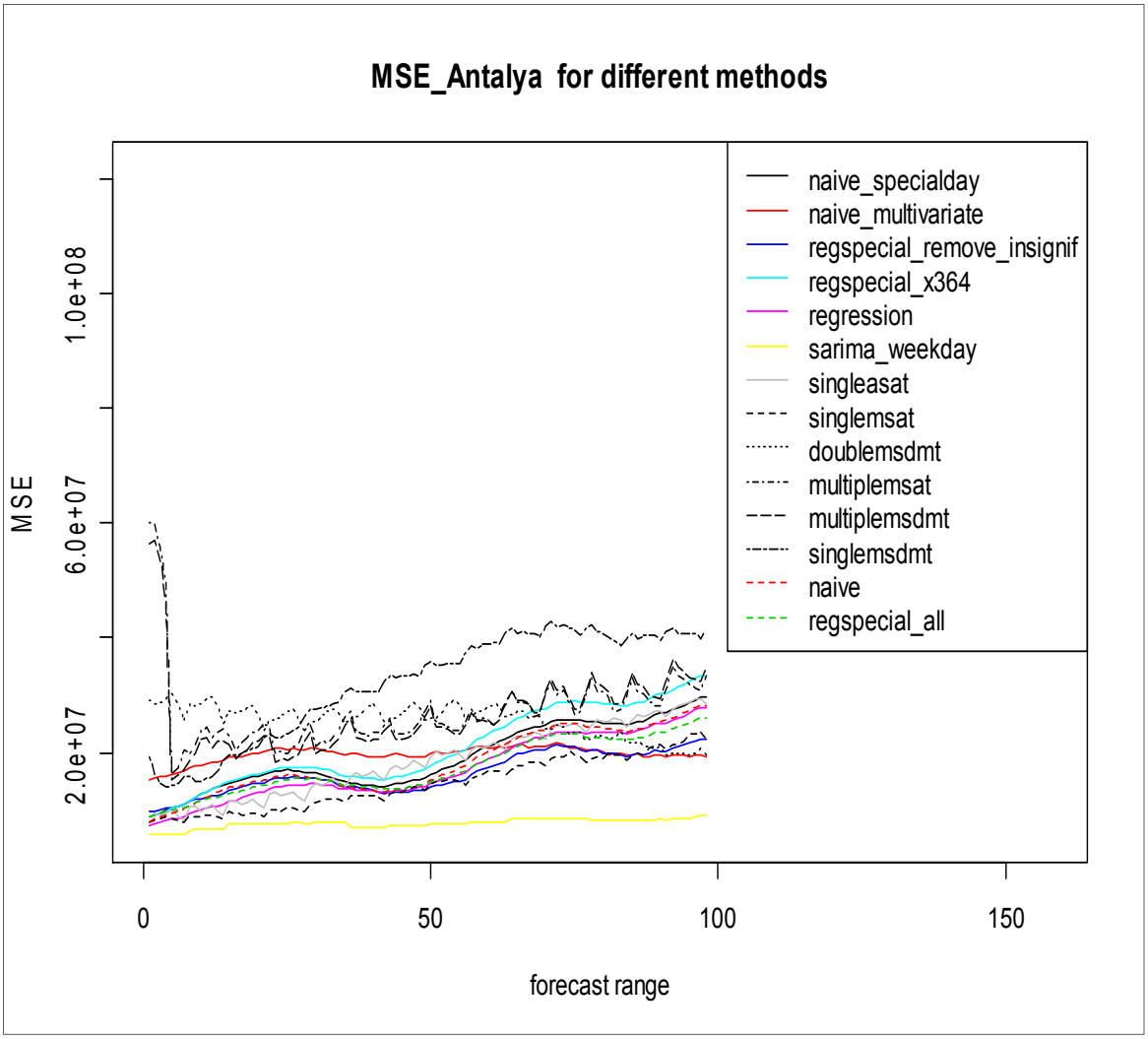

Figure 9.15. MSE plot for the Forecasting Methods for the Passenger Data

9.3.3. The Results for Bank Data

The results in Table 9.7 for the experiments for exponential smoothing methods show that for "Bank 1" daily cash withdrawal data, *simpleexposmooth* gives the smallest forecast errors, and that *singleasdmt, doubleasdmt, multipleasdmt* and *singleasmt* models also give relatively good forecast results. For the "Bank 2" daily cash deposits data, *simpleexposmooth* gives the smallest MAPE while *doubleasdmt, multipleasdmt, singleasdmt* methods give larger but still relatively small forecast errors.

Table 9.7. MAPE for Exponential Smoothing Methods for the Bank Data

Exponential Smoothing Methods	**Bank 1 mean MAPE**	**Bank 2 mean MAPE**	**Exponential Smoothing Methods**	**Bank 1 mean MAPE**	**Bank 2 mean MAPE**
doubleasat	0.82115	0.50078	multipleasdmt	0.62532	0.39821
doubleasatauto	0.82182	0.50151	multipleasdmtauto	0.64205	0.47558
doubleasdmt	0.62493	0.39936	multipleasmtauto	0.84095	0.53281
doubleasdmtauto	0.63991	0.43693	multiplemsat	0.82355	0.57617
doubleasmtauto	0.84094	0.52815	multiplemsatauto	0.82372	0.56297
doublemsat	0.82357	0.57153	multiplemsdmt	0.64555	0.43296
doublemsatauto	0.82368	0.56160	multiplemsdmtauto	0.64846	0.43259
doublemsdmt	0.64186	0.43310	multiplemsmtauto	0.84849	0.62874
doublemsdmtauto	0.64019	0.43259	simpleexposmooth	**0.51651**	**0.31099**
doublemsmtauto	0.83923	0.58602	singleasdmt	0.52905	0.39873
holtsexposmooth	0.91624	0.46595	singleasmt	0.54337	0.52826
multipleasat	0.83951	0.49932	singlemsdmt	0.57013	0.42966
multipleasatauto	0.82048	0.49937	singlemsmt	0.63589	0.59001

Table 9.8. MAPE for Forecasting Methods for the Bank Data

Methods	**Bank 1 mean MAPE**	**Bank 2 mean MAPE**
Singleasat	0.47173	0.34988
Singlemsat	0.42195	0.34873
Simpleexposmooth	0.48889	0.26245
Singleasdmt	0.44928	0.37965
Naive	0.41715	0.37844
Regression	**0.34773**	**0.22467**
Sarima_weekday	0.40559	0.39987

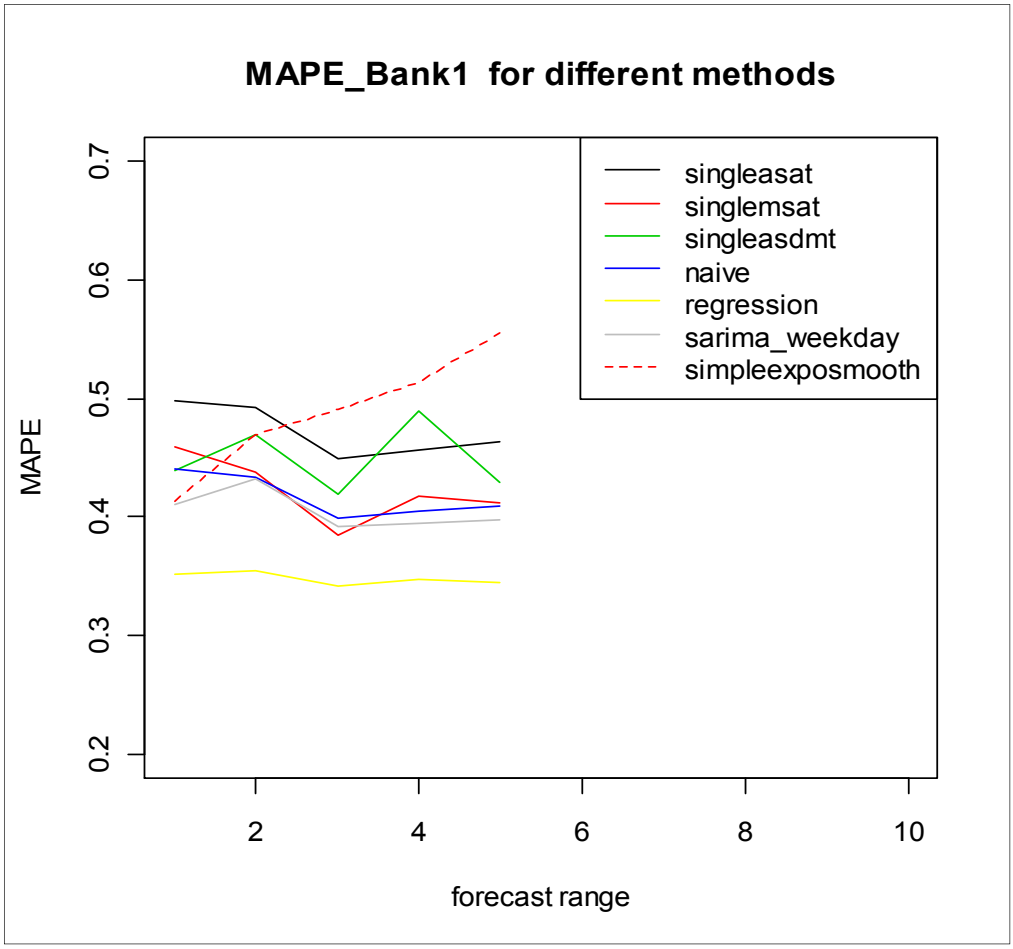

Figure 9.16. MAPE plot for the Forecasting Methods for the Bank 1 Data

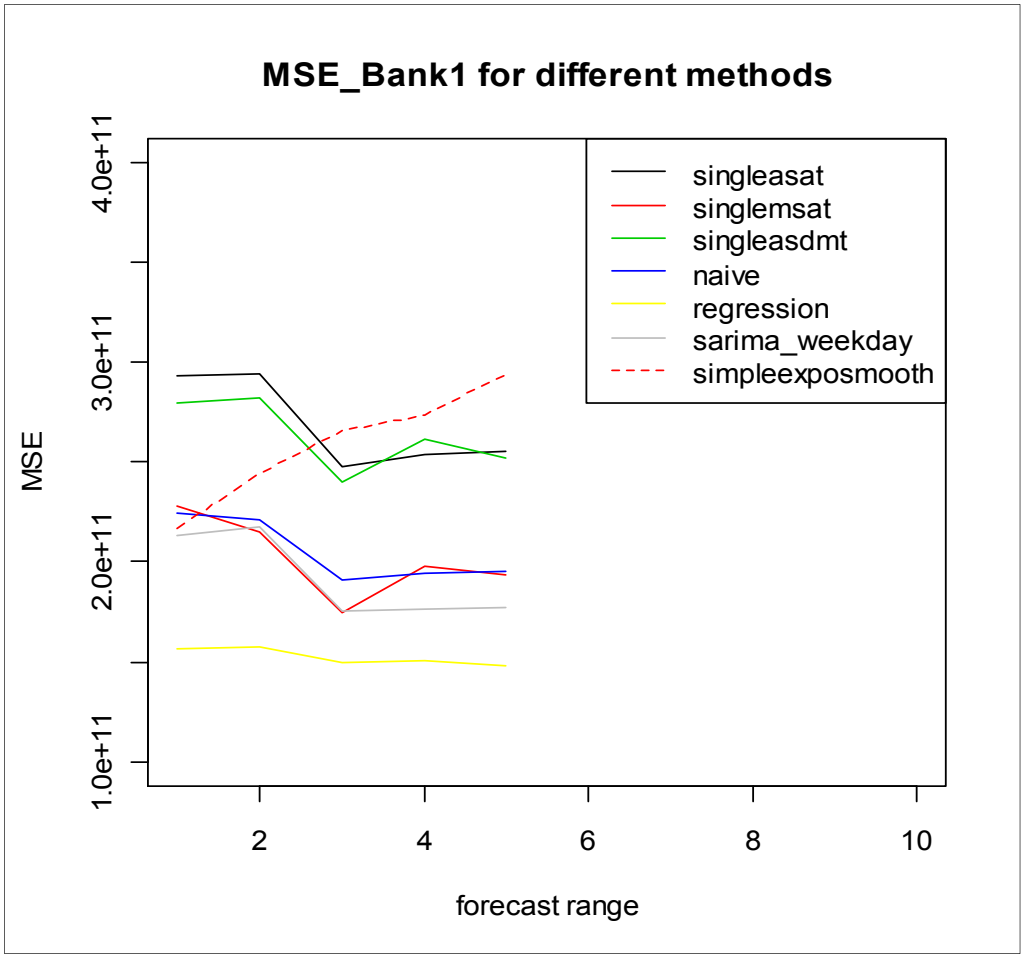

Figure 9.17. MSE plot for the Forecasting Methods for the Bank 1 Data

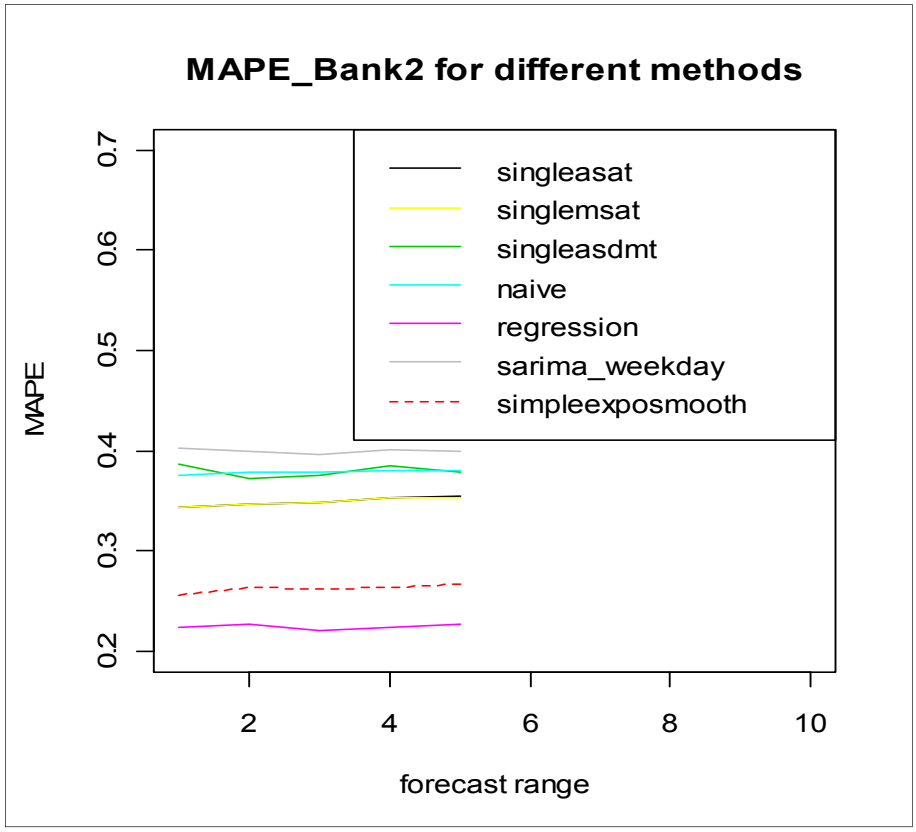

Figure 9.18. MAPE plot for the Forecasting Methods for the Bank 2 Data

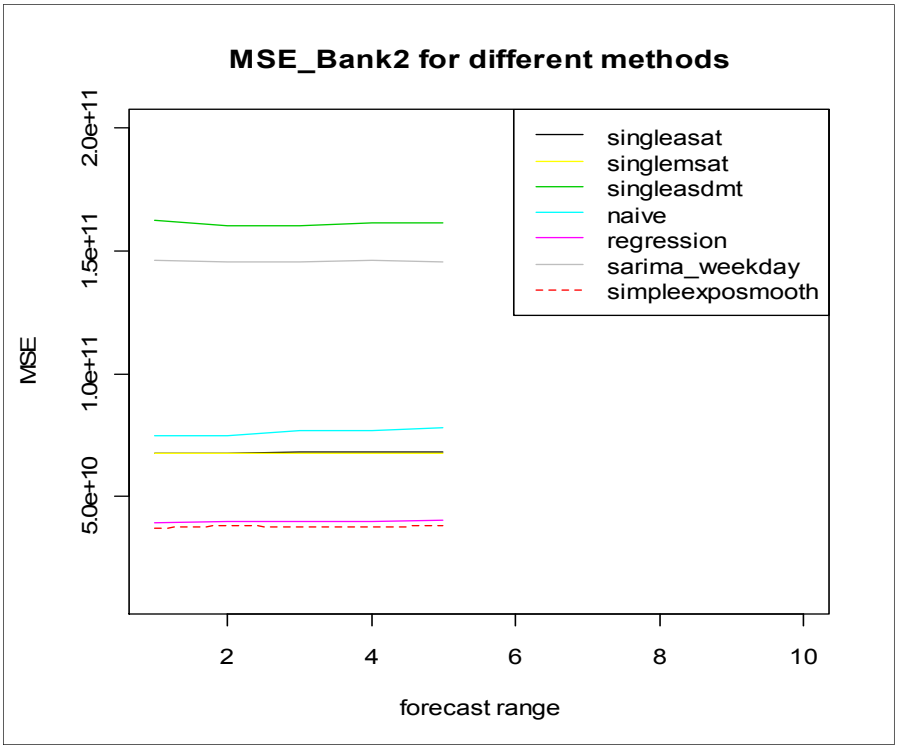

Figure 9.19. MSE plot for the Forecasting Methods for the Bank 2 Data

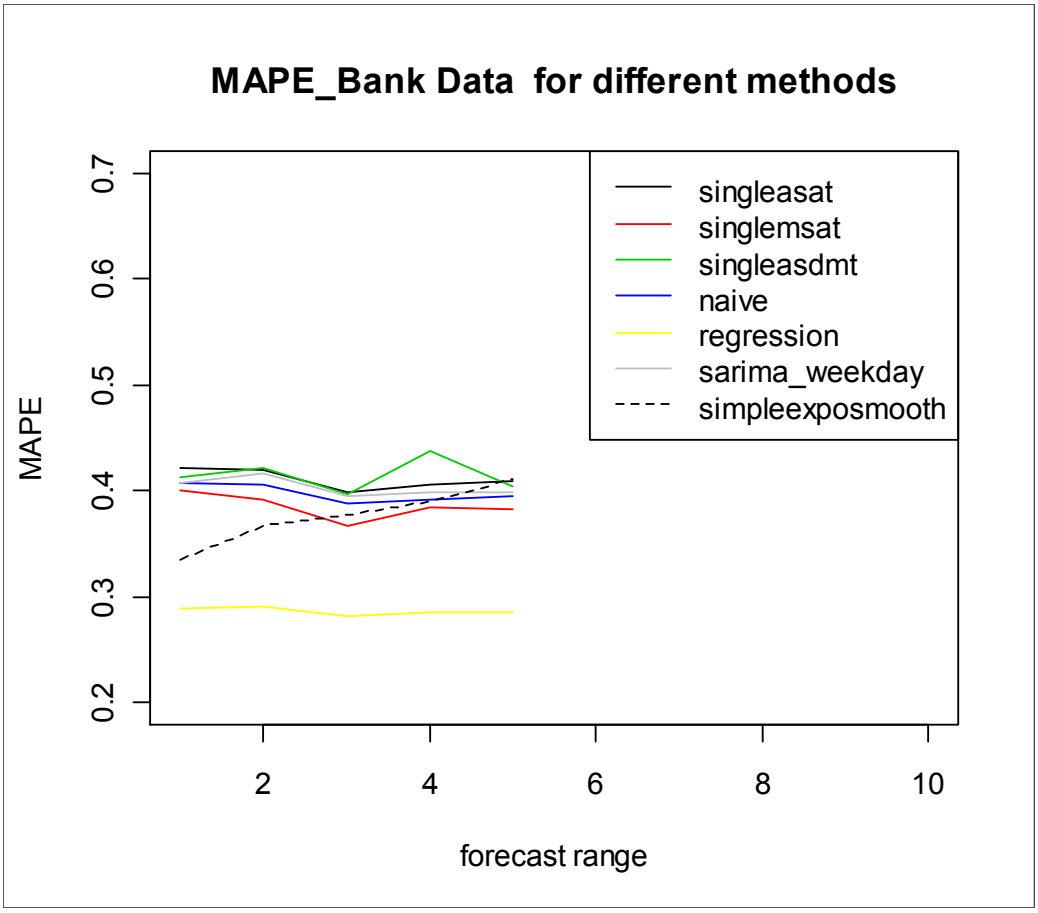

Figure 9.20. MAPE plot for days for the Forecasting Methods for the Bank Data

Overall, the *regression* method is well ahead of the other forecasting methods, although there are autocorrelation problems between errors and the regression model assumptions are not fulfilled. The reason for obtaining good forecasting results is that it determines the coefficients of lagged values used in forecasting future values according to the calculated significances and uses the least squares method and a change ratio. This leads to better results when we use lagged values with coefficients determined according to the fitted regression model. Also, the proper determination of suitable or correlated regression variables plays an important role. In the *regression* method, 364, 371, 357 and 728 days lagged values were used along with a logarithmic transformation. When we do the historic forecasting experiments for more than one year, the regression method gives a MAPE of 0.36760 for Bank 1 and 0.25048 for Bank 2. We see that in longer periods of simulation MAPE values are nearly the same.

Once again, exponential smoothing methods are not as successful as regression methods. The bank data length is a few months longer than three years. For short time series, multiple seasonal exponential smoothing methods are more disadvantageous, since these methods require an extra year to find weekly seasonal indices during one year. The reason is that there is not an automatic application in the "R" program finding weekly seasonal indices updated in each week in the seasonal indices initialization period. Furthermore, the simple exponential smoothing method is not outperformed by seasonal Holt-Winter's exponential smoothing methods. The reason is that the mean of the time series is nearly stable except from some outliers, and the time series data do not have a clear yearly seasonality. However, when we decompose the time series data by extracting random or stochastic component and trend, it can be observed that there is a perceptible yearly seasonality with a slight increase in cash withdrawals and deposits during the summers.

The Bank 1 time series is more variable and max/min ratio is much higher than the max/min ratio of Bank 2 time series.

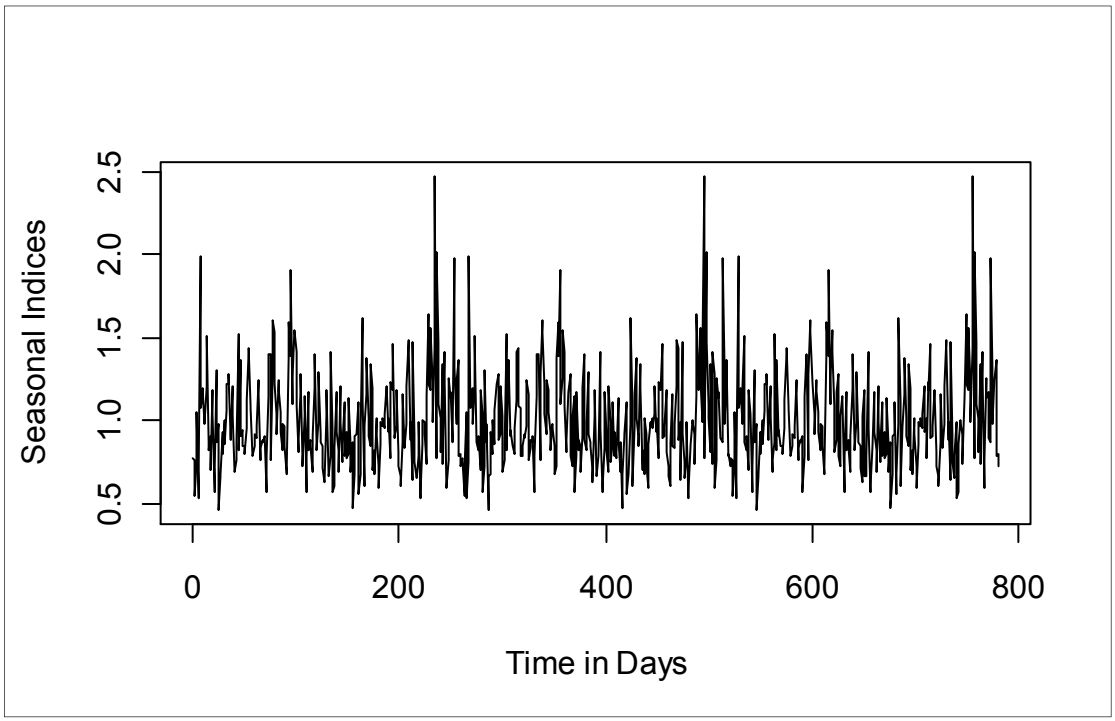

Figure 9.21. Cash withdrawal yearly seasonal indices plot of Bank 1

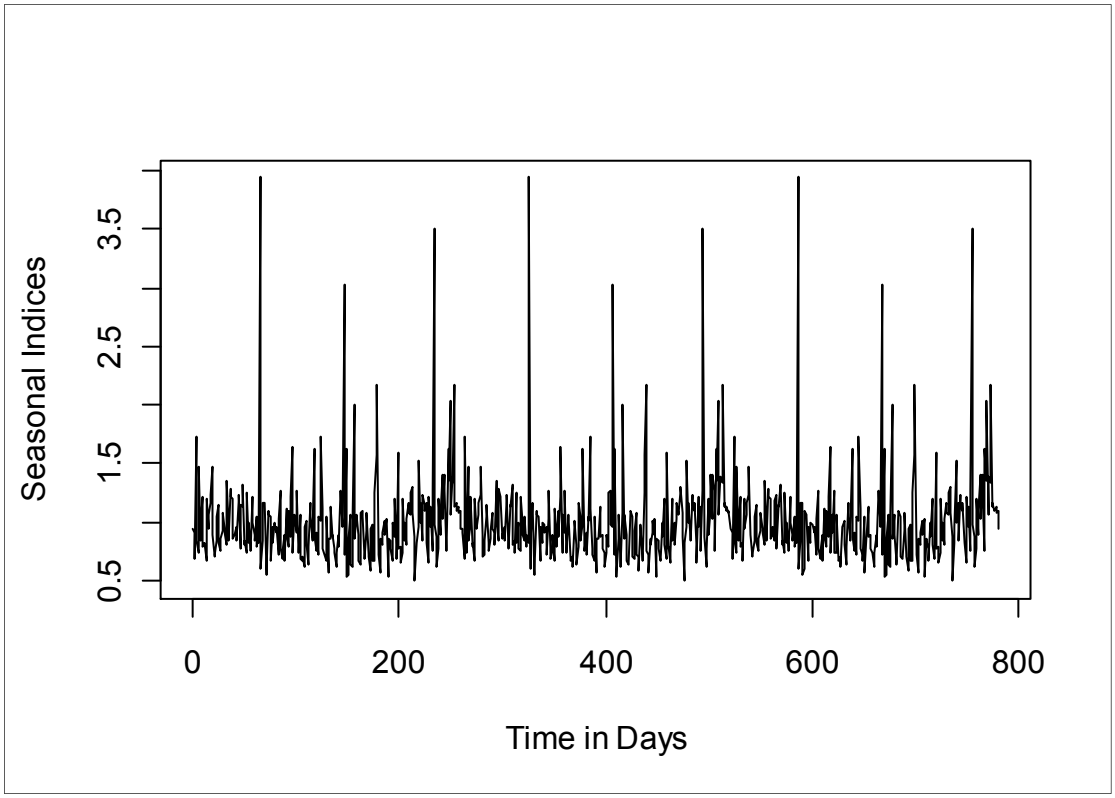

Figure 9.22. Cash deposit yearly seasonal indices plot of Bank 2

10. CONCLUSIONS

The aim of this thesis was to assess the performance of daily forecasting methods for several different daily time series. Therefore, three different daily and seasonal time series data made up of flight, passenger and bank data were analysed and different daily forecasting methods were evaluated and compared for different forecasting lead times. In this thesis, the forecasting methods which are applied to daily and seasonal time series data are developed by consideration of some details about daily and seasonal data such as special days' effects, appropriate lag numbers for yearly and weekly periodicities and change ratio of the mean of the days in the same periods of two sequential years. These several different point of views provided satisfactory forecasting results for the different data sets analyzed.

According to our experimental results, regression function using lagged and logarithmic regressors and dummy variables for special days works fine for time series which have an observable special days effect and the regression function using lagged and logarithmic regressors without dummy variables outperforms other forecasting methods for the time series data in which special days do not differ from normal days. These log linear and lagged variable regression models estimate coefficients using ordinary least squares regression and the autocorrelation between errors is tried to be eliminated by using appropriate lagged dependent variables. Also, by taking the logarithm of variables, probable multiplicative relationships between variables are transformed into additive relationships.

Double or multiple seasonal exponential smoothing methods do not show better performance for flight, passenger and bank data. For flight and passenger data, yearly seasonality is more distinct than weekly seasonality. Also, weekly seasonal indices change in different seasons of a year, and thus double seasonal models are not successful in time series which have strong yearly seasonality. For multiple seasonal exponential smoothing methods, weekly seasonal indices are updated every week, but they are influenced by yearly seasonal indices as both yearly and weekly seasonal indices are used for obtaining

smoothed values. For bank data both yearly and weekly seasonality are not very outstanding.

Although exponential smoothing methods do not give poor results in general, their performance is not as good as that of the regression methods. The method applied to day of the week series composed of Mondays, Tuesdays, Wednesdays, Thursdays, Fridays, Saturdays and Sundays generally does not produce smaller MAPE's than other forecasting methods, because the order of autoregression, seasonal autoregression, moving average, seasonal moving average and differences are required to be updated when the characteristics of time series change. This situation designates the good performance of log-linear and lagged regression models in forecasting daily and seasonal time series.

It is also important to note that, for the time series studied and for most of the forecasting models, the forecast errors do not increase dramatically when forecasting horizon increases. The reason is that 364 days of periodicity was used. This provides a good forecast performance for longer forecast lead times if there is no sudden or unexpected changes in time series and if the time series has a strong yearly seasonality.

It is important that a forecasting method gives smaller forecasting errors. However, it is also important that the model is easy to use. One of the disadvantages of the *sarima_weekday* method is that order and season parameters require to be updated periodically when existing parameters do not fit the changing characteristic of time series data. In addition, at the beginning there is a need for adjusting model parameters giving minimum autocorrelation values between sequential errors.

Additional processing for special days in naive and regression methods requires good calendar information about national and religious holidays. When we think about coding effort, *naive* method and single seasonal exponential smoothing codes are the easiest methods to apply. These methods can be applied when there is limited time and rough information about future values is needed.

Finally, it can be seen that seasonal daily time series forecast values can be considerably successful and relative errors can be fairly small when we apply smart and reasonable forecasting methods with special applications explained in this thesis.

APPENDIX A: R CODES

A.1. R Codes For Naive Methods

A.1.1. Naive( ) Function Codes

```
naive<- function(dv,n.ahead,nwindow=14,firstdayofdata=as.Date("2000-01-01")){
# dv ... vector with daily data
# nwindow ... length of window used for calculating the factor
# n.ahead ... length of returned daily prediction vector
N <- length(dv)
iv <- (N-nwindow+1):N
factor <- sum(dv[iv])/sum(dv[iv-364])
forecast <- factor*dv[((N+1):(N+n.ahead))-364]
return(forecast)
}
```

A.1.2. Naive_specialday( ) Function Codes

```
naive_specialday<- function(dv,n.ahead,nwindow=14,firstdayofdata=as.Date("2005-01-
01"),specialdate=c(paste("12",23:31,sep="-"),"01-01","01-02","12-08","08-15","10-26","11-01")){
# dv ... vector with daily data
# nwindow ... length of window used for calculating the factor
# n.ahead ... length of returned daily prediction vector
N <- length(dv)
names(dv) <- firstdayofdata+(0:(N-1))
iv <- (N-nwindow+1):N
factor <- sum(dv[iv])/sum(dv[iv-364])
forecast <- factor*dv[((N+1):(N+n.ahead))-364]
names(forecast) <- firstdayofdata+(N:(N+n.ahead-1))
forecast364<-factor*dv[((N+1):(N+n.ahead))-364]
        for(j in 1:length(specialdate)){
        for(k in 1:n.ahead){
if(substr(names(forecast),6,10)[k]!=specialdate[j]& substr(names(forecast364),6,10)[k]==specialdate[j])
forecast[k]<-factor*
dv[paste(substr(names(forecast364)[k],1,4),substr(names(forecast),6,7)[k],substr(names(forecast),9,10)[k],se
p="-")]
```

```
}}
if(length(specialdate) > 0 ){
thisyear <- as.numeric(substr(names(forecast)[1],1,4))
# print(paste(thisyear,specialdate[1],sep="-"))
# print(specialday)
        for(i in 1:length(specialdate)){
specialday <- as.Date(paste(thisyear,specialdate[i],sep="-"))
if(specialday < as.Date(names(forecast)[1]))
specialday <- as.Date(paste(thisyear+1,specialdate[i],sep="-"))
if(as.Date(names(forecast)[n.ahead])>=specialday){
forecast[as.character(specialday)] <- factor*dv[paste((as.numeric(substr(specialday,1,4))-
1),specialdate[i],sep="-")]
} } }
return(forecast)
}
```

A.1.3. Naive_weighted_average( ) Function Codes

```
naive_weighted_average <-
function(coefficient,dv,new1=MSEdata,n.ahead,nwindow=14,firstdayofdata=as.Date("2005-01-
01"),specialdate=c(paste("12",23:31,sep="-"),"01-01","01-02","12-08","08-15","10-26","11-01")){
factor357<-coefficient[1]
factor371<-coefficient[2]
factor364<-coefficient[3]
# dv ... vector with daily data
# nwindow ... length of window used for calculating the factor
# n.ahead ... length of returned daily prediction vector
N <- length(dv)
names(dv) <- firstdayofdata+(0:(N-1))
iv <- (N-nwindow+1):N
factor <- sum(dv[iv])/sum(dv[iv-364])
forecast <- factor*dv[((N+1):(N+n.ahead))-364]
names(forecast) <- firstdayofdata+(N:(N+n.ahead-1))
forecast364<-factor*dv[((N+1):(N+n.ahead))-364]
dv357<-factor357*dv[((N+1):(N+n.ahead))-357]
dv371<-factor371*dv[((N+1):(N+n.ahead))-371]
factor357<-1-(factor364+factor371)
combinedforecast<-(factor364*forecast)+factor*(dv357+dv371)
names(combinedforecast)<- firstdayofdata+(N:(N+n.ahead-1))
```

```
        for(j in 1:length(specialdate)){
        for(k in 1:n.ahead){
if(substr(names(forecast),6,10)[k]!=specialdate[j]& substr(names(forecast364),6,10)[k]==specialdate[j])
forecast[k]<-factor*
dv[paste(substr(names(forecast364)[k],1,4),substr(names(forecast),6,7)[k],substr(names(forecast),9,10)[k],se
p="-")]
}}
if(length(specialdate) > 0 ){
thisyear <- as.numeric(substr(names(forecast)[1],1,4))
# print(paste(thisyear,specialdate[1],sep="-"))
# print(specialday)
        for(i in 1:length(specialdate)){
specialday <- as.Date(paste(thisyear,specialdate[i],sep="-"))
if(specialday < as.Date(names(forecast)[1]))
specialday <- as.Date(paste(thisyear+1,specialdate[i],sep="-"))
if(as.Date(names(forecast)[n.ahead])>=specialday){
forecast[as.character(specialday)] <- factor*dv[paste((as.numeric(substr(specialday,1,4))-
1),specialdate[i],sep="-")]
}}}
if(n.ahead==98){
return(combinedforecast)
}
if(n.ahead>98){
return(mean((as.vector(combinedforecast)-new1)^2))
}
}
```

A.2. R Codes For Regression Methods

A.2.1. Lastyeardate( ) Function Codes

```
lastyeardate <- function(inpdate,dyear=-1){
#returns same date in an earlier year (
#dyear can be -1, -2, or -3
#29.2.2008 dyear=-1 returns 1.3.2007
#1.3.2008 dyear=-1 returns also 1.3.2007
year <- as.numeric(substr(as.character(inpdate),1,4))
month <- as.numeric(substr(as.character(inpdate),6,7))
day <- as.numeric(substr(as.character(inpdate),9,10))
```

```
lvschalttag <- (month==2&day==29)
month[lvschalttag] <- 3
day[lvschalttag] <- 1
as.Date(ISOdate(year+dyear,month,day))
}
```

A.2.2. Getlastsamedate( ) Function Codes

```
getlastsamedate <- function(dv,firstdayofdata=as.Date("2005-01-01"),dyear=-1){
#dv ... vector with daily data (with firstdayofdata)
N <- length(dv)
names(dv) <- firstdayofdata+(0:(N-1))
newdates <- lastyeardate(names(dv),dyear=dyear)
dv[as.character(newdates[newdates>=firstdayofdata])]
}
```

A.2.3. Maketsregressiondata( ) Function Codes

```
maketsregressiondata <- function(dv,n.ahead=n.ahead,nwindow=14,firstdayofdata=as.Date("2005-01-
01"),specialdate=c(paste("12",23:31,sep="-"),"01-01","01-02","01-03"),backmax=728){
# make regression data for create n.ahead forecasts
N <- length(dv)
names(dv) <- firstdayofdata+(0:(N-1))
y <- dv[-(1:(backmax+nwindow+n.ahead-1))]
nregr <- length(y)
x364 <- dv[as.character(as.Date(names(y))-364)]
        for(j in 1:length(specialdate)){
        for(i in 1:length(y)){
if(substr(names(y[i]),6,10)==specialdate[j])
x364[i]<-dv[paste((as.numeric(substr(names(y[i]),1,4))-1),specialdate[j],sep="-")]
}}
        for(j in 1:length(specialdate)){
        for(i in 1:length(y)){
if(substr(names(y[i]),6,10)!=specialdate[j]& substr(names(x364[i]),6,10)==specialdate[j])
x364[i]<-dv[paste((as.numeric(substr(names(y[i]),1,4))-1), substr(names(y[i]),6,10),sep="-")]
}}
x357 <- dv[as.character(as.Date(names(y))-357)]
if(backmax >= 371) x371 <- dv[as.character(as.Date(names(y))-371)]
if(backmax >= 728) x728 <- dv[as.character(as.Date(names(y))-728)]
```

```
#last year same date vector
x365 <- getlastsamedate(dv,firstdayofdata=firstdayofdata,dyear=-1)
x365 <- x365[-(1:(length(x365)-length(y)))]
if(backmax >= 730){
x730 <- getlastsamedate(dv,firstdayofdata=firstdayofdata,dyear=-2)
x730 <- x730[-(1:(length(x730)-length(y)))]
}
wratio <- rep(-99,nregr)
mathis <- rep(-99,nregr)
malast <- rep(-99,nregr)
        for(m in 1:nregr){
wratio[m]<-mean(dv[(N-nwindow+1-nregr+m):(N-nregr+m)])/ mean(dv[(N-nwindow+1-nregr+m-364):(N-
nregr+m-364)])
mathis[m]<-mean(dv[(N-nwindow+1-nregr+m):(N-nregr+m)])
malast[m]<-mean(dv[(N-nwindow+1-nregr+m-364):(N-nregr+m-364)])
}
sondertag <- rep(0,nregr)
if(length(specialdate) > 0 ){
monat <- as.numeric(substr(names(y),6,7))
tag <- as.numeric(substr(names(y),9,10))
        for(i in 1:length(specialdate)){
sondertag <-
sondertag+as.numeric(monat==as.numeric(substr(specialdate[i],1,2))&tag==as.numeric(substr(specialdate[i],
4,5)))
}}
if(backmax < 371){ data.frame(y,x357,x364,x365,wratio,sondertag,mathis,malast)
}else if(backmax >= 371 & backmax<728){
data.frame(y,x357,x364,x365,x371,wratio,sondertag,mathis,malast)
}else if(backmax >= 728& backmax<730){
data.frame(y,x357,x364,x365,x371,x728,wratio,sondertag,mathis,malast)
}else if(backmax >= 730){ data.frame(y,x357,x364,x365,x371,x728,x730,wratio,sondertag,mathis,malast)
}}
```

A.2.4. Datanahead( ) Function Codes

```
datanahead <- function(dv,n.ahead=n.ahead,firstdayofdata=as.Date("2005-01-
01"),specialdate=c(paste("12",23:31,sep="-"),"01-01","01-02","01-03"),nwindow=14){
# dv ... vector with daily data
# n.ahead ... length of returned daily prediction vector
```

```
N <- length(dv)
names(dv) <- firstdayofdata+(0:(N-1))
forecast <- dv[((N+1):(N+n.ahead))-364]
names(forecast) <- firstdayofdata+(N:(N+n.ahead-1))
forecast364<-dv[((N+1):(N+n.ahead))-364]
names(forecast364)<- firstdayofdata+((N-364):(N+n.ahead-1-364))
wrationahead<-rep(0,n.ahead)
mathisnahead<-rep(0,n.ahead)
malastnahead<-rep(0,n.ahead)
datax365nahead<-rep(0,n.ahead)
datax730nahead<-rep(0,n.ahead)
datax357nahead <- dv[((N+1):(N+n.ahead))-357]
datax371nahead<- dv[((N+1):(N+n.ahead))-371]
datax728nahead<-dv[((N+1):(N+n.ahead))-728]
if(length(specialdate) > 0 ){
        for(j in 1:length(specialdate)){
        for(k in 1:n.ahead){
datax365nahead[k]<-dv[paste(as.numeric(substr(names(forecast[k]),1,4))-1,
substr(names(forecast[k]),6,10),sep= "-")]
datax730nahead[k]<-dv[paste(as.numeric(substr(names(forecast[k]),1,4))-2,
substr(names(forecast[k]),6,10),sep= "-")]
if(substr(names(forecast[k]),6,10)!=specialdate[j]& substr(names(forecast364[k]),6,10)==specialdate[j])
forecast[k]<-
dv[paste(substr(names(forecast364[k]),1,4),substr(names(forecast[k]),6,7),substr(names(forecast[k]),9,10),se
p="-")]}}
thisyear <- as.numeric(substr(names(forecast)[1],1,4))
#print(paste(thisyear,specialdate[1],sep="-"))
#print(specialday)
        for(i in 1:length(specialdate)){
specialday <- as.Date(paste(thisyear,specialdate[i],sep="-"))
if(specialday < as.Date(names(forecast[1])))
specialday <- as.Date(paste(thisyear+1,specialdate[i],sep="-"))
if(as.Date(names(forecast[n.ahead]))>=specialday){
forecast[as.character(specialday)] <- dv[paste((as.numeric(substr(specialday,1,4))-1),specialdate[i],sep="-")]
}}}
        for(m in 1:n.ahead){
wrationahead[m]<- mean(dv[(N-nwindow+1):(N)])/ mean(dv[(N-nwindow+1-364):(N-364)])
mathisnahead[m]<-mean(dv[(N-nwindow+1):(N)])
malastnahead[m]<-mean(dv[(N-nwindow+1-364):(N-364)])
}
```

```
sondertag <- rep(0,n.ahead)
if(length(specialdate) > 0 ){
monat <- as.numeric(substr(names(forecast),6,7))
tag <- as.numeric(substr(names(forecast),9,10))
        for(i in 1:length(specialdate)){
sondertag <-
sondertag+as.numeric(monat==as.numeric(substr(specialdate[i],1,2))&tag==as.numeric(substr(specialdate[i],
4,5)))
}}
data.frame(forecast,datax365nahead,datax371nahead,datax357nahead,datax730nahead,wrationahead,mathisn
ahead,malastnahead,datax728nahead,sondertag)
}
```

A.2.5. Regspecial_remove_insignif( ) Function Codes

```
regspecial_remove_insignif<-function(dv,simdays,n.ahead){
forecastm<-matrix(rep(0,(simdays*n.ahead)),ncol=n.ahead)
realm<-matrix(rep(0,(simdays*n.ahead)),ncol=n.ahead)
L<-length(dv)
        for(k in 1:simdays){
mydata<-maketsregressiondata(dv[1:(L-n.ahead-simdays+k)],n.ahead)
reg0<-lm(I(log(y)) ~ I(log(x364)) + I(log(x365)) + I(log(x357)) + I(log(x371)) + I(log(x728)) +
I(log(wratio)),data=mydata[mydata$sondertag==0,])
reg1<-lm(I(log(y)) ~ I(log(x364)) + I(log(x365)) + I(log(x357)) + I(log(x371)) + I(log(x728)) +
I(log(wratio)),data=mydata[mydata$sondertag==1,])
par0<-reg0$rank
reg0coef<-as.vector(reg0$coefficient)
n.aheadvector<-datanahead(dv[1:(L-n.ahead-simdays+k)],n.ahead)
datax364.nahead<-n.aheadvector[,1]
datax365.nahead<-n.aheadvector[,2]
datax371.nahead<-n.aheadvector[,3]
datax357.nahead<-n.aheadvector[,4]
datax730.nahead<-n.aheadvector[,5]
wratio.nahead<-n.aheadvector[,6]
mathis.nahead<-n.aheadvector[,7]
malast.nahead<-n.aheadvector[,8]
datax728.nahead<-n.aheadvector[,9]
sondertag.nahead<-n.aheadvector[,10]
        for(s in 1:n.ahead){
```

```
if(sondertag.nahead[s]==0){
if((summary(reg0)$coefficient)[(par0)*3+1]>=0.1){
reg0<-lm(I(log(y)) ~ 0+I(log(x364)) + I(log(x365)) + I(log(x357)) + I(log(x371)) + I(log(x728)) +
I(log(wratio)),data=mydata[mydata$sondertag==0,])
forecastm[k,s]<-((datax364.nahead[s])^(reg0coef[1])) * ((datax365.nahead[s])^(reg0coef[2]))
*((datax357.nahead[s])^(reg0coef[3])) *
((datax371.nahead[s])^(reg0coef[4]))*((datax728.nahead[s])^(reg0coef[5])) *
((wratio.nahead[s])^(reg0coef[6]))
}
if((summary(reg0)$coefficient)[((par0-1)*3)+1]>=0.1){
reg0<-lm(I(log(y)) ~ I(log(x365)) + I(log(x357)) + I(log(x371)) + I(log(x728)) +
I(log(wratio)),data=mydata[mydata$sondertag==0,])
forecastm[k,s]<-exp(reg0coef[1])*((datax365.nahead[s])^(reg0coef[2]))
*((datax357.nahead[s])^(reg0coef[3])) *
((datax371.nahead[s])^(reg0coef[4]))*((datax728.nahead[s])^(reg0coef[5])) *
((wratio.nahead[s])^(reg0coef[6]))
}
if((summary(reg0)$coefficient)[((par0-1)*3)+2]>=0.1){
reg0<-lm(I(log(y)) ~ I(log(x364)) +I(log(x357)) + I(log(x371)) + I(log(x728)) +
I(log(wratio)),data=mydata[mydata$sondertag==0,])
forecastm[k,s]<- exp(reg0coef[1])*((datax364.nahead[s])^(reg0coef[2]))
*((datax357.nahead[s])^(reg0coef[3])) *
((datax371.nahead[s])^(reg0coef[4]))*((datax728.nahead[s])^(reg0coef[5])) *
((wratio.nahead[s])^(reg0coef[6]))
}
if((summary(reg0)$coefficient)[((par0-1)*3)+3]>=0.1){
reg0<-lm(I(log(y)) ~ I(log(x364)) + I(log(x365)) +I(log(x371)) + I(log(x728)) +
I(log(wratio)),data=mydata[mydata$sondertag==0,])
forecastm[k,s]<- exp(reg0coef[1])*((datax364.nahead[s])^(reg0coef[2]))
*((datax365.nahead[s])^(reg0coef[3]))
*((datax371.nahead[s])^(reg0coef[4]))*((datax728.nahead[s])^(reg0coef[5])) *
((wratio.nahead[s])^(reg0coef[6]))
}
if((summary(reg0)$coefficient)[((par0-1)*3)+4]>=0.1){
reg0<-lm(I(log(y)) ~ I(log(x364)) + I(log(x365)) +I(log(x357)) + I(log(x728)) +
I(log(wratio)),data=mydata[mydata$sondertag==0,])
forecastm[k,s]<- exp(reg0coef[1])*((datax364.nahead[s])^(reg0coef[2]))
*((datax365.nahead[s])^(reg0coef[3]))
*((datax357.nahead[s])^(reg0coef[4]))*((datax728.nahead[s])^(reg0coef[5])) *
((wratio.nahead[s])^(reg0coef[6]))
```

```
}
if((summary(reg0)$coefficient)[((par0-1)*3)+5]>=0.1){
reg0<-lm(I(log(y)) ~ I(log(x364)) + I(log(x365)) +I(log(x357)) + I(log(x728)) +
I(log(wratio)),data=mydata[mydata$sondertag==0,])
forecastm[k,s]<- exp(reg0coef[1])*((datax364.nahead[s])^(reg0coef[2]))
*((datax365.nahead[s])^(reg0coef[3]))
((datax357.nahead[s])^(reg0coef[4]))*((datax371.nahead[s])^(reg0coef[5])) *
((wratio.nahead[s])^(reg0coef[6]))
}
if((summary(reg0)$coefficient)[((par0-1)*3)+6]>=0.1){
reg0<-lm(I(log(y)) ~ I(log(x364)) + I(log(x365)) +I(log(x357)) + I(log(x371)) +I(log(x728)) +
I(log(wratio)),data=mydata[mydata$sondertag==0,])
forecastm[k,s]<- exp(reg0coef[1])*((datax364.nahead[s])^(reg0coef[2]))
*((datax365.nahead[s])^(reg0coef[3]))
((datax357.nahead[s])^(reg0coef[4]))*((datax371.nahead[s])^(reg0coef[5])) *
((datax728.nahead[s])^(reg0coef[6]))
}
else{
reg0<-lm(I(log(y)) ~ I(log(x364)) +I(log(x365))+I(log(x357)) + I(log(x371)) + I(log(x728)) +
I(log(wratio)),data=mydata[mydata$sondertag==0,])
forecastm[k,s]<-exp(reg0coef[1])*((datax364.nahead[s])^(reg0coef[2])) *
((datax365.nahead[s])^(reg0coef[3])) *((datax357.nahead[s])^(reg0coef[4])) *
((datax371.nahead[s])^(reg0coef[5]))*((datax728.nahead[s])^(reg0coef[6])) *
((wratio.nahead[s])^(reg0coef[7]))
}
}
if(sondertag.nahead[s]==1){
reg1<-lm(I(log(y)) ~ I(log(x364)) + I(log(wratio)),data=mydata[mydata$sondertag==1,])
reg1coef<-as.vector(reg1$coefficient)
forecastm[k,s]<- exp(reg1coef[1]) *((datax364.nahead[s])^(reg1coef[2])) *
((wratio.nahead[s])^(reg1coef[3]))
}
}}
        for(a in 1:simdays){
        for(b in 1:n.ahead){
realm[a,b]<-dv[L-n.ahead-simdays+a+b]
}}
return(forecastm)
}
```

A.2.6. Regspecial_all Function Codes

```
regspecial_all<-function(dv,simdays,n.ahead){
forecastm<-matrix(rep(0,(simdays*n.ahead)),ncol=n.ahead)
realm<-matrix(rep(0,(simdays*n.ahead)),ncol=n.ahead)
L<-length(dv)
        for(k in 1:simdays){
mydata<-maketsregressiondata(dv[1:(L-n.ahead-simdays+k)],n.ahead)
reg0<-lm(I(log(y)) ~ I(log(x364)) + I(log(x365)) + I(log(x357)) + I(log(x371)) + I(log(x728)) +
I(log(mathis)) + I(log(malast)),data=mydata[mydata$sondertag==0,])
reg1<-lm(I(log(y)) ~ I(log(x364)) + I(log(x357)) + I(log(x371)) + I(log(x728)) + I(log(mathis)) +
I(log(malast)),data=mydata[mydata$sondertag==1,])
par0<-reg0$rank
par1<-reg1$rank
reg0coef<-as.vector(reg0$coefficient)
reg1coef<-as.vector(reg1$coefficient)
n.aheadvector<-datanahead(dv[1:(L-n.ahead-simdays+k)],n.ahead)
datax364.nahead<-n.aheadvector[,1]
datax365.nahead<-n.aheadvector[,2]
datax371.nahead<-n.aheadvector[,3]
datax357.nahead<-n.aheadvector[,4]
datax730.nahead<-n.aheadvector[,5]
wratio.nahead<-n.aheadvector[,6]
mathis.nahead<-n.aheadvector[,7]
malast.nahead<-n.aheadvector[,8]
datax728.nahead<-n.aheadvector[,9]
sondertag.nahead<-n.aheadvector[,10]
        for(s in 1:n.ahead){
if(sondertag.nahead[s]==0)
forecastm[k,s]<-exp(reg0coef[1])*((datax364.nahead[s])^(reg0coef[2])) *
((datax365.nahead[s])^(reg0coef[3])) *((datax357.nahead[s])^(reg0coef[4])) *
((datax371.nahead[s])^(reg0coef[5]))*((datax728.nahead[s])^(reg0coef[6])) *
((mathis.nahead[s])^(reg0coef[7]))* ((malast.nahead[s])^(reg0coef[8]))
if(sondertag.nahead[s]==1)
forecastm[k,s]<-exp(reg1coef[1])*((datax364.nahead[s])^(reg1coef[2]))
*((datax357.nahead[s])^(reg1coef[3])) *
((datax371.nahead[s])^(reg1coef[4]))*((datax728.nahead[s])^(reg1coef[5])) *
((mathis.nahead[s])^(reg1coef[6]))* ((malast.nahead[s])^(reg1coef[7]))
}}
```

```
return(forecastm)
}
```

A.2.7. Regspecial_x364( ) Function Codes

```
regspecial_x364<-function(dv,simdays,n.ahead){
forecastm<-matrix(rep(0,(simdays*n.ahead)),ncol=n.ahead)
realm<-matrix(rep(0,(simdays*n.ahead)),ncol=n.ahead)
L<-length(dv)
        for(k in 1:simdays){
mydata<-maketsregressiondata(dv[1:(L-n.ahead-simdays+k)],n.ahead)
reg0<-lm(I(log(y)) ~ I(log(x364)) + I(log(mathis)) + I(log(malast)),data=mydata[mydata$sondertag==0,])
reg1<-lm(I(log(y)) ~ I(log(x364)) + I(log(mathis)) + I(log(malast)),data=mydata[mydata$sondertag==1,])
par0<-reg0$rank
par1<-reg1$rank
reg0coef<-as.vector(reg0$coefficient)
reg1coef<-as.vector(reg1$coefficient)
n.aheadvector<-datanahead(dv[1:(L-n.ahead-simdays+k)],n.ahead)
datax364.nahead<-n.aheadvector[,1]
wratio.nahead<-n.aheadvector[,6]
mathis.nahead<-n.aheadvector[,7]
malast.nahead<-n.aheadvector[,8]
sondertag.nahead<-n.aheadvector[,10]
        for(s in 1:n.ahead){
if(sondertag.nahead[s]==0)
forecastm[k,s]<-exp(reg0coef[1])*((datax364.nahead[s])^(reg0coef[2])) *
((mathis.nahead[s])^(reg0coef[3]))* ((malast.nahead[s])^(reg0coef[4]))
if(sondertag.nahead[s]==1)
forecastm[k,s]<-exp(reg1coef[1])*((datax364.nahead[s])^(reg1coef[2]))
*((mathis.nahead[s])^(reg1coef[3]))* ((malast.nahead[s])^(reg1coef[4]))
}}
return(forecastm)
}
```

A.2.8. Regression( ) Function Codes

```
regression<-function(data,n.ahead,n.window,begincut,endcut){
ratio<-vector(mode= "logical")
a<-vector(mode="logical")
```

```
b<-vector(mode="logical")
coef<-matrix(rep(0,((endcut-n.ahead)*6)),ncol=6)
matr<-matrix(rep(0,((endcut-n.ahead)*n.ahead)),ncol=n.ahead)
        for(j in 1:(endcut-n.ahead)){
y<-data[1:(length(data)-(endcut-j))]
x364<-data[1:(length(y)+n.ahead-364)]
x371<-data[1:(length(y)+n.ahead-371)]
x357<-data[1:(length(y)+n.ahead-357)]
x728<-data[1:(length(y)+n.ahead-728)]
        for(i in (n.window+364):(length(y))){
a[i]<-mean(as.vector(data)[(i-n.window+1):(i)])/ mean(as.vector(data)[(i-n.window+1):(i)-364])}
ratio<-a[(n.window+364):(length(y))]
shortestlength<-min(length(y),length(x364),length(x371),length(x357),length(x728),length(ratio))
newy<-y[-(1:(length(y)+begincut-shortestlength))]
newx364<-x364[-(1:(length(x364)+begincut-shortestlength))]
newx371<-x371[-(1:(length(x371)+begincut-shortestlength))]
newx357<-x357[-(1:(length(x357)+begincut-shortestlength))]
newratio<-ratio[-(1:(length(ratio)+begincut-shortestlength))]
newx728<-x728[-(1:begincut)]
newwy<-newy[-(1:n.ahead)]
newwx364<-newx364[1:(length(newx364)-n.ahead)]
newwx371<-newx371[1:(length(newx371)-n.ahead)]
newwx357<-newx357[1:(length(newx357)-n.ahead)]
newwx728<-newx728[1:(length(newx728)-n.ahead)]
newwratio<-newratio[-(1:n.ahead)]
d<-data.frame(newwy,newwx364,newwx371,newwx357,newwx728,newwratio)
reg<-lm(log(d[,1])~log(d[,2])+log(d[,3])+log(d[,4])+log(d[,5])+log(d[,6]))
coef[j,]<- as.vector(reg$coefficient)
b<-coef[j,]
        for(k in 1:n.ahead){
matr[j,k]<-b[1]+b[2]*log(newx364[(length(newx364)-n.ahead+k)])+b[3]*log(newx371[(length(newx371)-
n.ahead+k)])+b[4]*log(newx357[(length(newx357)-n.ahead+k)])+b[5]*log(newx728[(length(newx728)-
n.ahead+k)])+b[6]*log(newwratio[length(newwratio)])
}}
return(exp(matr))
}
```

A.3. R Codes For Exponential Smoothing Methods

A.3.1. Single Seasonal Exponential Smoothing Methods

Single Additive Seasonal Additive Trend Holt Winters

```
singleasat<-function(lambda,data,n.ahead,initdatayear){
#singleasat function uses single additive seasonal and additive trend Holt Winters R function and returns optimum alpha, beta and gamma parameters when n.ahead is 0 and returns from one step to n.ahead step ahead forecast results when n.ahead is greater than 0.
#data is the time series data that is going to be used in forecasting.
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
init<-as.vector(data)[1:(initdatayear*364)]
ts<-ts(init,start=1,frequency=364)
h<-HoltWinters(ts,
seasonal="additive",start.periods=4,optim.start=c(alpha=lambda1,beta=lambda2,gamma=lambda3),optim.co
ntrol=list(maxit=10000))
if(n.ahead==0){
return(as.vector(c(h$alpha,h$beta,h$gamma)))}
else{
return(predict(h,n.ahead=n.ahead))}
}
```

Single Multiplicative Seasonal Additive Trend Holt Winters

```
singlemsat<-function(lambda,data,n.ahead,initdatayear){
#singlemsat function uses single multiplicative seasonal and additive trend Holt Winters R function and returns optimum alpha, beta and gamma parameters when n.ahead is 0 and returns from one step to n.ahead step ahead forecast results when n.ahead is greater than 0.
#data is the time series data that is going to be used in forecasting.
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
init<-as.vector(data)[1:(initdatayear*364)]
ts<-ts(init,start=1,frequency=364)
```

```
h<-HoltWinters(ts,
seasonal="multiplicative",start.periods=4,optim.start=c(alpha=lambda1,beta=lambda2,gamma=lambda3),opti
m.control=list(maxit=10000))
if(n.ahead==0){
return(as.vector(c(h$alpha,h$beta,h$gamma)))}
else{
return(predict(h,n.ahead=n.ahead))}
}
```

Single Additive Seasonal Multiplicative Damped Trend Exponential Smoothing

```
singleasdmt<-function(lambda,data,n.ahead,initdatayear){
 lambda1<-lambda[1]
 lambda2<-lambda[2]
 lambda3<-lambda[3]
 lambda4<-lambda[4]
onestepfnum=length(data)-initdatayear*364-1
 init<-as.vector(data)[1:(initdatayear*364)]
 ts<-ts(init,start=1,frequency=364)
d<-decompose(ts,type="additive",filter=NULL)
 t<-d$trend
s<-(as.vector(d$seasonal))[1:364]
 l<-c(1:(initdatayear*364))
 g<-lm(as.vector(t)~l)
 a<-vector(mode="logical")
 b<-vector(mode="logical")
onestepf<-vector(mode="logical")
e<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*(as.vector(data)[(initdatayear*364+1)]-s[1])+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda4
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda4
s[365]<-lambda3*(as.vector(data)[(initdatayear*364+1)]-a[1])+(1-lambda3)*s[1]
onestepf[1]<-(a[1]*b[1]^lambda4)+s[2]
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])-s[m+1])+(1-lambda1)*(a[m]*b[m]^lambda4)
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*b[m]^lambda4
```

```
s[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])-a[m+1])+(1-lambda3)*s[m+1]
onestepf[m+1]<-(a[m+1]*b[m+1]^lambda4)+s[m+2]
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
        for(z in 1:(i+1)){
e[z]<-(lambda4^(z))}
se<-sum(e)
onestepf[i+onestepfnum]<-(a[onestepfnum]*b[onestepfnum]^(se))+s[i+onestepfnum+1-364]}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Single Additive Seasonal Multiplicative Trend Exponential Smoothing

```
singleasmt<-function(lambda,data,n.ahead,initdatayear){
 lambda1<-lambda[1]
 lambda2<-lambda[2]
 lambda3<-lambda[3]
onestepfnum=length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
 ts<-ts(init,start=1,frequency=364)
d<-decompose(ts,type="additive",filter=NULL)
 t<-d$trend
s<-(as.vector(d$seasonal))[1:364]
 l<-c(1:(initdatayear*364))
 g<-lm(as.vector(t)~l)
 a<-vector(mode="logical")
 b<-vector(mode="logical")
onestepf<-vector(mode="logical")
 initbeta0<-as.vector(g$coefficients)[1]
 initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*(as.vector(data)[(initdatayear*364+1)]-s[1])+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
 b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))
s[365]<-lambda3*(as.vector(data)[(initdatayear*364+1)]-a[1])+(1-lambda3)*s[1]
onestepf[1]<-(a[1]*b[1]^1)+s[2]
```

```
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])-s[m+1])+(1-lambda1)*(a[m]*b[m])
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*b[m]
s[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])-a[m+1])+(1-lambda3)*s[m+1]
onestepf[m+1]<-(a[m+1]*b[m+1]^1)+s[m+2]
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
 onestepf[i+onestepfnum]<-(a[onestepfnum]*b[onestepfnum]^(i+1))+s[i+onestepfnum+1-364]}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Single Multiplicative Seasonal Multiplicative Damped Trend Exponential Smoothing

```
singlemsdmt<-function(lambda,data,n.ahead,initdatayear){
 lambda1<-lambda[1]
 lambda2<-lambda[2]
 lambda3<-lambda[3]
 lambda4<-lambda[4]
onestepfnum=length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
 ts<-ts(init,start=1,frequency=364)
d<-decompose(ts,type="multiplicative",filter=NULL)
 t<-d$trend
s<-(as.vector(d$seasonal))[1:364]
 l<-c(1:(initdatayear*364))
 g<-lm(as.vector(t)~l)
 a<-vector(mode="logical")
 b<-vector(mode="logical")
onestepf<-vector(mode="logical")
e<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[(initdatayear*364+1)])/s[1])+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda4
```

```
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda4
s[365]<-lambda3*((as.vector(data)[(initdatayear*364+1)])/a[1])+(1-lambda3)*s[1]
onestepf[1]<-(a[1]*b[1]^lambda4)*s[2]
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])/s[m+1])+(1-lambda1)*(a[m]*b[m]^lambda4)
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*b[m]^lambda4
s[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])/a[m+1])+(1-lambda3)*s[m+1]
onestepf[m+1]<-(a[m+1]*b[m+1]^lambda4)*s[m+2]
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
        for(z in 1:(i+1)){
e[z]<-(lambda4^(z))}
se<-sum(e)
onestepf[i+onestepfnum]<-((a[onestepfnum]*b[onestepfnum]^(se))*s[i+onestepfnum+1-364])}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Single Multiplicative Seasonal Multiplicative Trend Exponential Smoothing

```
singlemsmt<-function(lambda,data,n.ahead,initdatayear){
onestepfnum=length(data)-initdatayear*364-1
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
init<-as.vector(data)[1:(initdatayear*364)]
ts<-ts(init,start=1,frequency=364)
d<-decompose(ts,type="multiplicative",filter=NULL)
t<-d$trend
s<-(as.vector(d$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
```

```
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[(initdatayear*364+1)])/s[1])+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))
s[365]<-lambda3*((as.vector(data)[(initdatayear*364+1)])/a[1])+(1-lambda3)*s[1]
onestepf[1]<-(a[1]*b[1]^1)*s[2]
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])/s[m+1])+(1-lambda1)*(a[m]*b[m])
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*b[m]
s[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])/a[m+1])+(1-lambda3)*s[m+1]
onestepf[m+1]<-(a[m+1]*b[m+1]^1)*s[m+2]
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
onestepf[i+onestepfnum]<-((a[onestepfnum]*b[onestepfnum]^(i+1))*s[i+onestepfnum+1-364])}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

A.3.2. Double Seasonal Exponential Smoothing Methods

Double Additive Seasonal Additive Trend Exponential Smoothing

```
doubleasat<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
onestepfnum=length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="additive",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="additive",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
```

```
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])-(s1[1]+s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
b[1]<-lambda2*(a[1]-(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*initbeta1
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])-(a[1]+s2[1]))+(1-lambda3)*s1[1]
onestepf[1]<-(a[1]+b[1]*1)+s1[2]+s2[2]
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])-(s1[m+1]+s2[m+1]))+(1-
lambda1)*(a[m]+b[m])
b[m+1]<-lambda2*(a[m+1]-a[m])+(1-lambda2)*b[m]
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])-(a[m+1]+s2[m+1]))+(1-
lambda3)*s1[m+1]
onestepf[m+1]<-(a[m+1]+b[m+1]*1)+s1[m+2]+s2[m+2]}
if(onestepfnum>363){
init2<-as.vector(data)[1:((initdatayear+1)*364)]
ts22<-ts(init2,start=1,frequency=7)
d22<-decompose(ts22,type="additive",filter=NULL)
s22<-(as.vector(d22$seasonal))[1:364]
s2[365:728]<-s22
        for(n in 363:(onestepfnum-1)){
a[n+1]<-lambda1*((as.vector(data)[initdatayear*364+1+n])-(s1[n+1]+s2[n+1]))+(1-lambda1)*(a[n]+b[n])
b[n+1]<-lambda2*(a[n+1]-a[n])+(1-lambda2)*b[n]
s1[364+n+1]<-lambda3*((as.vector(data)[initdatayear*364+1+n])-(a[n+1]+s2[n+1]))+(1-lambda3)*s1[n+1]
onestepf[n+1]<-(a[n+1]+b[n+1]*1)+s1[n+2]+s2[n+2]}
}
if(onestepfnum>727){
init3<-as.vector(data)[1:((initdatayear+2)*364)]
ts23<-ts(init3,start=1,frequency=7)
d23<-decompose(ts23,type="additive",filter=NULL)
s23<-(as.vector(d23$seasonal))[1:364]
s2[729:1092]<-s23
        for(p in 727:(onestepfnum-1)){
a[p+1]<-lambda1*((as.vector(data)[initdatayear*364+1+p])-(s1[p+1]+s2[p+1]))+(1-lambda1)*(a[p]+b[p])
b[p+1]<-lambda2*(a[p+1]-a[p])+(1-lambda2)*b[p]
```

```
s1[364+p+1]<-lambda3*((as.vector(data)[initdatayear*364+1+p])-(a[p+1]+s2[p+1]))+(1-lambda3)*s1[p+1]
onestepf[p+1]<-(a[p+1]+b[p+1]*1)+s1[p+2]+s2[p+2]}
}
if(onestepfnum>1091){
init4<-as.vector(data)[1:((initdatayear+3)*364)]
ts24<-ts(init4,start=1,frequency=7)
d24<-decompose(ts24,type="additive",filter=NULL)
s24<-(as.vector(d24$seasonal))[1:364]
s2[1093:1456]<-s24
        for(k in 1091:(onestepfnum-1)){
a[k+1]<-lambda1*((as.vector(data)[initdatayear*364+1+k])-(s1[k+1]+s2[k+1]))+(1-lambda1)*(a[k]+b[k])
b[k+1]<-lambda2*(a[k+1]-a[k])+(1-lambda2)*b[k]
s1[364+k+1]<-lambda3*((as.vector(data)[initdatayear*364+1+k])-(a[k+1]+s2[k+1]))+(1-lambda3)*s1[k+1]
onestepf[k+1]<-(a[k+1]+b[k+1]*1)+s1[k+2]+s2[k+2]}
}
if(onestepfnum>1455){
init5<-as.vector(data)[1:((initdatayear+4)*364)]
ts25<-ts(init5,start=1,frequency=7)
d25<-decompose(ts25,type="additive",filter=NULL)
s25<-(as.vector(d25$seasonal))[1:364]
s2[1457:1820]<-s25
        for(r in 1455:(onestepfnum-1)){
a[r+1]<-lambda1*((as.vector(data)[initdatayear*364+1+r])-(s1[r+1]+s2[r+1]))+(1-lambda1)*(a[r]+b[r])
b[r+1]<-lambda2*(a[r+1]-a[r])+(1-lambda2)*b[r]
s1[364+r+1]<-lambda3*((as.vector(data)[initdatayear*364+1+r])-(a[r+1]+s2[r+1]))+(1-lambda3)*s1[r+1]
onestepf[r+1]<-(a[r+1]+b[r+1]*1)+s1[r+2]+s2[r+2]}
}
if(onestepfnum>1819){
init6<-as.vector(data)[1:((initdatayear+5)*364)]
ts26<-ts(init6,start=1,frequency=7)
d26<-decompose(ts26,type="additive",filter=NULL)
s26<-(as.vector(d26$seasonal))[1:364]
s2[1821:2184]<-s26
        for(j in 1819:(onestepfnum-1)){
a[j+1]<-lambda1*((as.vector(data)[initdatayear*364+1+j])-(s1[j+1]+s2[j+1]))+(1-lambda1)*(a[j]+b[j])
b[j+1]<-lambda2*(a[j+1]-a[j])+(1-lambda2)*b[j]
s1[364+j+1]<-lambda3*((as.vector(data)[initdatayear*364+1+j])-(a[j+1]+s2[j+1]))+(1-lambda3)*s1[j+1]
onestepf[j+1]<-(a[j+1]+b[j+1]*1)+s1[j+2]+s2[j+2]}
}
if(onestepfnum>2183){
```

```
init7<-as.vector(data)[1:((initdatayear+6)*364)]
ts27<-ts(init7,start=1,frequency=7)
d27<-decompose(ts27,type="additive",filter=NULL)
s27<-(as.vector(d27$seasonal))[1:364]
s2[2185:2548]<-s27
        for(y in 2183:(onestepfnum-1)){
a[y+1]<-lambda1*((as.vector(data)[initdatayear*364+1+y])-(s1[y+1]+s2[y+1]))+(1-lambda1)*(a[y]+b[y])
b[y+1]<-lambda2*(a[y+1]-a[y])+(1-lambda2)*b[y]
s1[364+y+1]<-lambda3*((as.vector(data)[initdatayear*364+1+y])-(a[y+1]+s2[y+1]))+(1-lambda3)*s1[y+1]
onestepf[y+1]<-(a[y+1]+b[y+1]*1)+s1[y+2]+s2[y+2]}
}
if(onestepfnum>2547){
init8<-as.vector(data)[1:((initdatayear+7)*364)]
ts28<-ts(init8,start=1,frequency=7)
d28<-decompose(ts28,type="additive",filter=NULL)
s28<-(as.vector(d28$seasonal))[1:364]
s2[2549:2912]<-s28
        for(o in 2547:(onestepfnum-1)){
a[o+1]<-lambda1*((as.vector(data)[initdatayear*364+1+o])-(s1[o+1]+s2[o+1]))+(1-lambda1)*(a[o]+b[o])
b[o+1]<-lambda2*(a[o+1]-a[o])+(1-lambda2)*b[o]
s1[364+o+1]<-lambda3*((as.vector(data)[initdatayear*364+1+o])-(a[o+1]+s2[o+1]))+(1-lambda3)*s1[o+1]
onestepf[o+1]<-(a[o+1]+b[o+1]*1)+s1[o+2]+s2[o+2]}
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
onestepf[i+onestepfnum]<-(a[onestepfnum]+b[onestepfnum]*(i+1))+s1[i+onestepfnum+1-
364]+s2[i+onestepfnum+1-364]}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Double Additive Seasonal Additive Trend Exponential Smoothing With Autocorrelation Adjustment

```
doubleasatauto<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
```

```
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="additive",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="additive",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])-(s1[1]+s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
b[1]<-lambda2*(a[1]-(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*initbeta1
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])-(a[1]+s2[1]))+(1-lambda3)*s1[1]
onestepf[1]<-(a[1]+b[1]*1)+s1[2]+s2[2]+lambda4^1*( as.vector(data)[initdatayear*364+1]-
((a[1])+s1[2]+s2[2]))
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+m+1])-(s1[m+1]+s2[m+1]))+(1-
lambda1)*(a[m]+b[m])
b[m+1]<-lambda2*(a[m+1]-a[m])+(1-lambda2)*b[m]
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+m+1])-(a[m+1]+s2[m+1]))+(1-
lambda3)*s1[m+1]
onestepf[m+1]<-(a[m+1]+b[m+1]*1)+s1[m+2]+s2[m+2]+lambda4*(as.vector(data)[initdatayear*364+m+1]-
((a[m+1])+s1[m+2]+s2[m+2]))}
if(onestepfnum>363){
init2<-as.vector(data)[1:((initdatayear+1)*364)]
ts22<-ts(init2,start=1,frequency=7)
d22<-decompose(ts22,type="additive",filter=NULL)
s22<-(as.vector(d22$seasonal))[1:364]
s2[365:728]<-s22
        for(n in 363:(onestepfnum-1)){
a[n+1]<-lambda1*((as.vector(data)[initdatayear*364+n+1])-(s1[n+1]+s2[n+1]))+(1-lambda1)*(a[n]+b[n])
b[n+1]<-lambda2*(a[n+1]-a[n])+(1-lambda2)*b[n]
s1[364+n+1]<-lambda3*((as.vector(data)[initdatayear*364+n+1])-(a[n+1]+s2[n+1]))+(1-lambda3)*s1[n+1]
```

```
onestepf[n+1]<-(a[n+1]+b[n+1]*1)+s1[n+2]+s2[n+2]+lambda4*(as.vector(data)[initdatayear*364+n+1]-
((a[n+1])+s1[n+2]+s2[n+2]))}
}
if(onestepfnum>727){
init3<-as.vector(data)[1:((initdatayear+2)*364)]
ts23<-ts(init3,start=1,frequency=7)
d23<-decompose(ts23,type="additive",filter=NULL)
s23<-(as.vector(d23$seasonal))[1:364]
s2[729:1092]<-s23
        for(p in 727:(onestepfnum-1)){
a[p+1]<-lambda1*((as.vector(data)[initdatayear*364+p+1])-(s1[p+1]+s2[p+1]))+(1-lambda1)*(a[p]+b[p])
b[p+1]<-lambda2*(a[p+1]-a[p])+(1-lambda2)*b[p]
s1[364+p+1]<-lambda3*((as.vector(data)[initdatayear*364+p+1])-(a[p+1]+s2[p+1]))+(1-lambda3)*s1[p+1]
onestepf[p+1]<-(a[p+1]+b[p+1]*1)+s1[p+2]+s2[p+2]+lambda4*(as.vector(data)[initdatayear*364+p+1]-
((a[p+1])+s1[p+2]+s2[p+2]))}
}
if(onestepfnum>1091){
init4<-as.vector(data)[1:((initdatayear+3)*364)]
ts24<-ts(init4,start=1,frequency=7)
d24<-decompose(ts24,type="additive",filter=NULL)
s24<-(as.vector(d24$seasonal))[1:364]
s2[1093:1456]<-s24
for(k in 1091:(onestepfnum-1)){
a[k+1]<-lambda1*((as.vector(data)[initdatayear*364+k+1])-(s1[k+1]+s2[k+1]))+(1-lambda1)*(a[k]+b[k])
b[k+1]<-lambda2*(a[k+1]-a[k])+(1-lambda2)*b[k]
s1[364+k+1]<-lambda3*((as.vector(data)[initdatayear*364+k+1])-(a[k+1]+s2[k+1]))+(1-lambda3)*s1[k+1]
onestepf[k+1]<-(a[k+1]+b[k+1]*1)+s1[k+2]+s2[k+2]+lambda4*(as.vector(data)[initdatayear*364+k+1]-
((a[k+1])+s1[k+2]+s2[k+2]))}
}
if(onestepfnum>1455){
init5<-as.vector(data)[1:((initdatayear+4)*364)]
ts25<-ts(init5,start=1,frequency=7)
d25<-decompose(ts25,type="additive",filter=NULL)
s25<-(as.vector(d25$seasonal))[1:364]
s2[1457:1820]<-s25
        for(r in 1455:(onestepfnum-1)){
a[r+1]<-lambda1*((as.vector(data)[initdatayear*364+r+1])-(s1[r+1]+s2[r+1]))+(1-lambda1)*(a[r]+b[r])
b[r+1]<-lambda2*(a[r+1]-a[r])+(1-lambda2)*b[r]
s1[364+r+1]<-lambda3*((as.vector(data)[initdatayear*364+r+1])-(a[r+1]+s2[r+1]))+(1-lambda3)*s1[r+1]
```

```
onestepf[r+1]<-(a[r+1]+b[r+1]*1)+s1[r+2]+s2[r+2]+lambda4*(as.vector(data)[initdatayear*364+r+1]-
((a[r+1])+s1[r+2]+s2[r+2]))}
}
if(onestepfnum>1819){
init6<-as.vector(data)[1:((initdatayear+5)*364)]
ts26<-ts(init6,start=1,frequency=7)
d26<-decompose(ts26,type="additive",filter=NULL)
s26<-(as.vector(d26$seasonal))[1:364]
s2[1821:2184]<-s26
        for(j in 1819:(onestepfnum-1)){
a[j+1]<-lambda1*((as.vector(data)[initdatayear*364+j+1])-(s1[j+1]+s2[j+1]))+(1-lambda1)*(a[j]+b[j])
b[j+1]<-lambda2*(a[j+1]-a[j])+(1-lambda2)*b[j]
s1[364+j+1]<-lambda3*((as.vector(data)[initdatayear*364+j+1])-(a[j+1]+s2[j+1]))+(1-lambda3)*s1[j+1]
onestepf[j+1]<-(a[j+1]+b[j+1]*1)+s1[j+2]+s2[j+2]+lambda4*(as.vector(data)[initdatayear*364+j+1]-
((a[j+1])+s1[j+2]+s2[j+2]))}
}
if(onestepfnum>2183){
init7<-as.vector(data)[1:((initdatayear+6)*364)]
ts27<-ts(init7,start=1,frequency=7)
d27<-decompose(ts27,type="additive",filter=NULL)
s27<-(as.vector(d27$seasonal))[1:364]
s2[2185:2548]<-s27
        for(y in 2183:(onestepfnum-1)){
a[y+1]<-lambda1*((as.vector(data)[initdatayear*364+y+1])-(s1[y+1]+s2[y+1]))+(1-lambda1)*(a[y]+b[y])
b[y+1]<-lambda2*(a[y+1]-a[y])+(1-lambda2)*b[y]
s1[364+y+1]<-lambda3*((as.vector(data)[initdatayear*364+y+1])-(a[y+1]+s2[y+1]))+(1-lambda3)*s1[y+1]
onestepf[y+1]<-(a[y+1]+b[y+1]*1)+s1[y+2]+s2[y+2]+lambda4*(as.vector(data)[initdatayear*364+y+1]-
((a[y+1])+s1[y+2]+s2[y+2]))}
}
if(onestepfnum>2547){
init8<-as.vector(data)[1:((initdatayear+7)*364)]
ts28<-ts(init8,start=1,frequency=7)
d28<-decompose(ts28,type="additive",filter=NULL)
s28<-(as.vector(d28$seasonal))[1:364]
s2[2549:2912]<-s28
        for(o in 2547:(onestepfnum-1)){
a[o+1]<-lambda1*((as.vector(data)[initdatayear*364+o+1])-(s1[o+1]+s2[o+1]))+(1-lambda1)*(a[o]+b[o])
b[o+1]<-lambda2*(a[o+1]-a[o])+(1-lambda2)*b[o]
s1[364+o+1]<-lambda3*((as.vector(data)[initdatayear*364+o+1])-(a[o+1]+s2[o+1]))+(1-lambda3)*s1[o+1]
```

```
onestepf[o+1]<-(a[o+1]+b[o+1]*1)+s1[o+2]+s2[o+2]+lambda4*(as.vector(data)[initdatayear*364+o+1]-
((a[o+1])+s1[o+2]+s2[o+2]))}
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
onestepf[i+onestepfnum]<-(a[onestepfnum]+b[onestepfnum]*(i+1))+s1[i+onestepfnum+1-
364]+s2[i+onestepfnum+1-364]+lambda4^(i)*(as.vector(data)[initdatayear*364+onestepfnum+1]-
((a[onestepfnum-1]+b[onestepfnum-1])+s1[onestepfnum+1-364]+s2[onestepfnum+1-364]))}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Double Additive Seasonal Multiplicative Damped Trend Exponential Smoothing

```
doubleasdmt<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="additive",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="additive",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
e<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
```

```
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])-(s1[1]+s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda4
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda4
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])-(a[1]+s2[1]))+(1-lambda3)*s1[1]
onestepf[1]<-(a[1]*b[1]^lambda4)+s1[2]+s2[2]
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])-(s1[m+1]+s2[m+1]))+(1-
lambda1)*(a[m]*b[m]^lambda4)
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*b[m]^lambda4
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])-(a[m+1]+s2[m+1]))+(1-
lambda3)*s1[m+1]
onestepf[m+1]<-(a[m+1]*b[m+1]^lambda4)+s1[m+2]+s2[m+2]}
if(onestepfnum>363){
init2<-as.vector(data)[1:((initdatayear+1)*364)]
ts22<-ts(init2,start=1,frequency=7)
d22<-decompose(ts22,type="additive",filter=NULL)
s22<-(as.vector(d22$seasonal))[1:364]
s2[365:728]<-s22
        for(n in 363:(onestepfnum-1)){
a[n+1]<-lambda1*((as.vector(data)[initdatayear*364+1+n])-(s1[n+1]+s2[n+1]))+(1-
lambda1)*(a[n]*b[n]^lambda4)
b[n+1]<-lambda2*(a[n+1]/a[n])+(1-lambda2)*b[n]^lambda4
s1[364+n+1]<-lambda3*((as.vector(data)[initdatayear*364+1+n])-(a[n+1]+s2[n+1]))+(1-lambda3)*s1[n+1]
onestepf[n+1]<-(a[n+1]*b[n+1]^lambda4)+s1[n+2]+s2[n+2]}
}
if(onestepfnum>727){
init3<-as.vector(data)[1:((initdatayear+2)*364)]
ts23<-ts(init3,start=1,frequency=7)
d23<-decompose(ts23,type="additive",filter=NULL)
s23<-(as.vector(d23$seasonal))[1:364]
s2[729:1092]<-s23
        for(p in 727:(onestepfnum-1)){
a[p+1]<-lambda1*((as.vector(data)[initdatayear*364+1+p])-(s1[p+1]+s2[p+1]))+(1-
lambda1)*(a[p]*b[p]^lambda4)
b[p+1]<-lambda2*(a[p+1]/a[p])+(1-lambda2)*b[p]^lambda4
s1[364+p+1]<-lambda3*((as.vector(data)[initdatayear*364+1+p])-(a[p+1]+s2[p+1]))+(1-lambda3)*s1[p+1]
onestepf[p+1]<-(a[p+1]*b[p+1]^lambda4)+s1[p+2]+s2[p+2]}
}
```

```
if(onestepfnum>1091){
init4<-as.vector(data)[1:((initdatayear+3)*364)]
ts24<-ts(init4,start=1,frequency=7)
d24<-decompose(ts24,type="additive",filter=NULL)
s24<-(as.vector(d24$seasonal))[1:364]
s2[1093:1456]<-s24
        for(k in 1091:(onestepfnum-1)){
a[k+1]<-lambda1*((as.vector(data)[initdatayear*364+1+k])-(s1[k+1]+s2[k+1]))+(1-
lambda1)*(a[k]*b[k]^lambda4)
b[k+1]<-lambda2*(a[k+1]/a[k])+(1-lambda2)*b[k]^lambda4
s1[364+k+1]<-lambda3*((as.vector(data)[initdatayear*364+1+k])-(a[k+1]+s2[k+1]))+(1-lambda3)*s1[k+1]
onestepf[k+1]<-(a[k+1]*b[k+1]^lambda4)+s1[k+2]+s2[k+2]}
}
if(onestepfnum>1455){
init5<-as.vector(data)[1:((initdatayear+4)*364)]
ts25<-ts(init5,start=1,frequency=7)
d25<-decompose(ts25,type="additive",filter=NULL)
s25<-(as.vector(d25$seasonal))[1:364]
s2[1457:1820]<-s25
        for(r in 1455:(onestepfnum-1)){
a[r+1]<-lambda1*((as.vector(data)[initdatayear*364+1+r])-(s1[r+1]+s2[r+1]))+(1-
lambda1)*(a[r]*b[r]^lambda4)
b[r+1]<-lambda2*(a[r+1]/a[r])+(1-lambda2)*b[r]^lambda4
s1[364+r+1]<-lambda3*((as.vector(data)[initdatayear*364+1+r])-(a[r+1]+s2[r+1]))+(1-lambda3)*s1[r+1]
onestepf[r+1]<-(a[r+1]*b[r+1]^lambda4)+s1[r+2]+s2[r+2]}
}
if(onestepfnum>1819){
init6<-as.vector(data)[1:((initdatayear+5)*364)]
ts26<-ts(init6,start=1,frequency=7)
d26<-decompose(ts26,type="additive",filter=NULL)
s26<-(as.vector(d26$seasonal))[1:364]
s2[1821:2184]<-s26
        for(j in 1819:(onestepfnum-1)){
a[j+1]<-lambda1*((as.vector(data)[initdatayear*364+1+j])-(s1[j+1]+s2[j+1]))+(1-
lambda1)*(a[j]*b[j]^lambda4)
b[j+1]<-lambda2*(a[j+1]/a[j])+(1-lambda2)*b[j]^lambda4
s1[364+j+1]<-lambda3*((as.vector(data)[initdatayear*364+1+j])-(a[j+1]+s2[j+1]))+(1-lambda3)*s1[j+1]
onestepf[j+1]<-(a[j+1]*b[j+1]^lambda4)+s1[j+2]+s2[j+2]}
}
if(onestepfnum>2183){
```

```
init7<-as.vector(data)[1:((initdatayear+6)*364)]
ts27<-ts(init7,start=1,frequency=7)
d27<-decompose(ts27,type="additive",filter=NULL)
s27<-(as.vector(d27$seasonal))[1:364]
s2[2185:2548]<-s27
        for(y in 2183:(onestepfnum-1)){
a[y+1]<-lambda1*((as.vector(data)[initdatayear*364+1+y])-(s1[y+1]+s2[y+1]))+(1-
lambda1)*(a[y]*b[y]^lambda4)
b[y+1]<-lambda2*(a[y+1]/a[y])+(1-lambda2)*b[y]^lambda4
s1[364+y+1]<-lambda3*((as.vector(data)[initdatayear*364+1+y])-(a[y+1]+s2[y+1]))+(1-lambda3)*s1[y+1]
onestepf[y+1]<-(a[y+1]*b[y+1]^lambda4)+s1[y+2]+s2[y+2]}
}
if(onestepfnum>2547){
init8<-as.vector(data)[1:((initdatayear+7)*364)]
ts28<-ts(init8,start=1,frequency=7)
d28<-decompose(ts28,type="additive",filter=NULL)
s28<-(as.vector(d28$seasonal))[1:364]
s2[2549:2912]<-s28
        for(o in 2547:(onestepfnum-1)){
a[o+1]<-lambda1*((as.vector(data)[initdatayear*364+1+o])-(s1[o+1]+s2[o+1]))+(1-
lambda1)*(a[o]*b[o]^lambda4)
b[o+1]<-lambda2*(a[o+1]/a[o])+(1-lambda2)*b[o]^lambda4
s1[364+o+1]<-lambda3*((as.vector(data)[initdatayear*364+1+o])-(a[o+1]+s2[o+1]))+(1-lambda3)*s1[o+1]
onestepf[o+1]<-(a[o+1]*b[o+1]^lambda4)+s1[o+2]+s2[o+2]}
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
for(z in 1:(i+1)){
e[z]<-(lambda4^(z))}
se<-sum(e)
onestepf[i+onestepfnum]<-(a[onestepfnum]*b[onestepfnum]^(se))+s1[i+onestepfnum+1-
364]+s2[i+onestepfnum+1-364]}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Double Additive Seasonal Multiplicative Damped Trend Exponential Smoothing With Autocorrelation Adjustment

```
doubleasdmtauto<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
lambda5<-lambda[5]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="additive",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="additive",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
e<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])-(s1[1]+s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda5
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda5
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])-(a[1]+s2[1]))+(1-lambda3)*s1[1]
onestepf[1]<-(a[1]*b[1]^lambda5)+s1[2]+s2[2]+lambda4^1*( as.vector(data)[initdatayear*364+1]-
((a[1])+s1[2]+s2[2]))
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])-(s1[m+1]+s2[m+1]))+(1-
lambda1)*(a[m]*b[m]^lambda5)
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*b[m]^lambda5
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])-(a[m+1]+s2[m+1]))+(1-
lambda3)*s1[m+1]
```

```
onestepf[m+1]<-
(a[m+1]*b[m+1]^lambda5)+s1[m+2]+s2[m+2]+lambda4*(as.vector(data)[initdatayear*364+1+m]-
((a[m+1])+s1[m+2]+s2[m+2]))}
if(onestepfnum>363){
init2<-as.vector(data)[1:((initdatayear+1)*364)]
ts22<-ts(init2,start=1,frequency=7)
d22<-decompose(ts22,type="additive",filter=NULL)
s22<-(as.vector(d22$seasonal))[1:364]
s2[365:728]<-s22
        for(n in 363:(onestepfnum-1)){
a[n+1]<-lambda1*((as.vector(data)[initdatayear*364+1+n])-(s1[n+1]+s2[n+1]))+(1-
lambda1)*(a[n]*b[n]^lambda5)
b[n+1]<-lambda2*(a[n+1]/a[n])+(1-lambda2)*b[n]^lambda5
s1[364+n+1]<-lambda3*((as.vector(data)[initdatayear*364+1+n])-(a[n+1]+s2[n+1]))+(1-lambda3)*s1[n+1]
onestepf[n+1]<-
(a[n+1]*b[n+1]^lambda5)+s1[n+2]+s2[n+2]+lambda4*(as.vector(data)[initdatayear*364+1+n]-
((a[n+1])+s1[n+2]+s2[n+2]))}
}
if(onestepfnum>727){
init3<-as.vector(data)[1:((initdatayear+2)*364)]
ts23<-ts(init3,start=1,frequency=7)
d23<-decompose(ts23,type="additive",filter=NULL)
s23<-(as.vector(d23$seasonal))[1:364]
s2[729:1092]<-s23
        for(p in 727:(onestepfnum-1)){
a[p+1]<-lambda1*((as.vector(data)[initdatayear*364+1+p])-(s1[p+1]+s2[p+1]))+(1-
lambda1)*(a[p]*b[p]^lambda5)
b[p+1]<-lambda2*(a[p+1]/a[p])+(1-lambda2)*b[p]^lambda5
s1[364+p+1]<-lambda3*((as.vector(data)[initdatayear*364+1+p])-(a[p+1]+s2[p+1]))+(1-lambda3)*s1[p+1]
onestepf[p+1]<-
(a[p+1]*b[p+1]^lambda5)+s1[p+2]+s2[p+2]+lambda4*(as.vector(data)[initdatayear*364+1+p]-
((a[p+1])+s1[p+2]+s2[p+2]))}
}
if(onestepfnum>1091){
init4<-as.vector(data)[1:((initdatayear+3)*364)]
ts24<-ts(init4,start=1,frequency=7)
d24<-decompose(ts24,type="additive",filter=NULL)
s24<-(as.vector(d24$seasonal))[1:364]
s2[1093:1456]<-s24
        for(k in 1091:(onestepfnum-1)){
```

```
a[k+1]<-lambda1*((as.vector(data)[initdatayear*364+1+k])-(s1[k+1]+s2[k+1]))+(1-
lambda1)*(a[k]*b[k]^lambda5)
b[k+1]<-lambda2*(a[k+1]/a[k])+(1-lambda2)*b[k]^lambda5
s1[364+k+1]<-lambda3*((as.vector(data)[initdatayear*364+1+k])-(a[k+1]+s2[k+1]))+(1-lambda3)*s1[k+1]
onestepf[k+1]<-
(a[k+1]*b[k+1]^lambda5)+s1[k+2]+s2[k+2]+lambda4*(as.vector(data)[initdatayear*364+1+k]-
((a[k+1])+s1[k+2]+s2[k+2]))}
}
if(onestepfnum>1455){
init5<-as.vector(data)[1:((initdatayear+4)*364)]
ts25<-ts(init5,start=1,frequency=7)
d25<-decompose(ts25,type="additive",filter=NULL)
s25<-(as.vector(d25$seasonal))[1:364]
s2[1457:1820]<-s25
        for(r in 1455:(onestepfnum-1)){
a[r+1]<-lambda1*((as.vector(data)[initdatayear*364+1+r])-(s1[r+1]+s2[r+1]))+(1-
lambda1)*(a[r]*b[r]^lambda5)
b[r+1]<-lambda2*(a[r+1]/a[r])+(1-lambda2)*b[r]^lambda5
s1[364+r+1]<-lambda3*((as.vector(data)[initdatayear*364+1+r])-(a[r+1]+s2[r+1]))+(1-lambda3)*s1[r+1]
onestepf[r+1]<-
(a[r+1]*b[r+1]^lambda5)+s1[r+2]+s2[r+2]+lambda4*(as.vector(data)[initdatayear*364+1+r]-
((a[r+1])+s1[r+2]+s2[r+2]))}
}
if(onestepfnum>1819){
init6<-as.vector(data)[1:((initdatayear+5)*364)]
ts26<-ts(init6,start=1,frequency=7)
d26<-decompose(ts26,type="additive",filter=NULL)
s26<-(as.vector(d26$seasonal))[1:364]
s2[1821:2184]<-s26
        for(j in 1819:(onestepfnum-1)){
a[j+1]<-lambda1*((as.vector(data)[initdatayear*364+1+j])-(s1[j+1]+s2[j+1]))+(1-
lambda1)*(a[j]*b[j]^lambda5)
b[j+1]<-lambda2*(a[j+1]/a[j])+(1-lambda2)*b[j]^lambda5
s1[364+j+1]<-lambda3*((as.vector(data)[initdatayear*364+1+j])-(a[j+1]+s2[j+1]))+(1-lambda3)*s1[j+1]
onestepf[j+1]<-(a[j+1]*b[j+1]^lambda5)+s1[j+2]+s2[j+2]+lambda4*(as.vector(data)[initdatayear*364+1+j]-
((a[j+1])+s1[j+2]+s2[j+2]))}
}
if(onestepfnum>2183){
init7<-as.vector(data)[1:((initdatayear+6)*364)]
ts27<-ts(init7,start=1,frequency=7)
```

```
d27<-decompose(ts27,type="additive",filter=NULL)
s27<-(as.vector(d27$seasonal))[1:364]
s2[2185:2548]<-s27
        for(y in 2183:(onestepfnum-1)){
a[y+1]<-lambda1*((as.vector(data)[initdatayear*364+1+y])-(s1[y+1]+s2[y+1]))+(1-
lambda1)*(a[y]*b[y]^lambda5)
b[y+1]<-lambda2*(a[y+1]/a[y])+(1-lambda2)*b[y]^lambda5
s1[364+y+1]<-lambda3*((as.vector(data)[initdatayear*364+1+y])-(a[y+1]+s2[y+1]))+(1-lambda3)*s1[y+1]
onestepf[y+1]<-
(a[y+1]*b[y+1]^lambda5)+s1[y+2]+s2[y+2]+lambda4*(as.vector(data)[initdatayear*364+1+y]-
((a[y+1])+s1[y+2]+s2[y+2]))}
}
if(onestepfnum>2547){
init8<-as.vector(data)[1:((initdatayear+7)*364)]
ts28<-ts(init8,start=1,frequency=7)
d28<-decompose(ts28,type="additive",filter=NULL)
s28<-(as.vector(d28$seasonal))[1:364]
s2[2549:2912]<-s28
        for(o in 2547:(onestepfnum-1)){
a[o+1]<-lambda1*((as.vector(data)[initdatayear*364+1+o])-(s1[o+1]+s2[o+1]))+(1-
lambda1)*(a[o]*b[o]^lambda5)
b[o+1]<-lambda2*(a[o+1]/a[o])+(1-lambda2)*b[o]^lambda5
s1[364+o+1]<-lambda3*((as.vector(data)[initdatayear*364+1+o])-(a[o+1]+s2[o+1]))+(1-lambda3)*s1[o+1]
onestepf[o+1]<-
(a[o+1]*b[o+1]^lambda5)+s1[o+2]+s2[o+2]+lambda4*(as.vector(data)[initdatayear*364+1+o]-
((a[o+1])+s1[o+2]+s2[o+2]))}
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
        for(z in 1:(i+1)){
e[z]<-(lambda5^(z))}
se<-sum(e)
onestepf[i+onestepfnum]<-(a[onestepfnum]*b[onestepfnum]^(se))+s1[i+onestepfnum+1-
364]+s2[i+onestepfnum+1-364]+lambda4^(i)*(as.vector(data)[onestepfnum+initdatayear*364+1]-
((a[onestepfnum-1]*b[onestepfnum-1]^lambda5)+s1[onestepfnum+1-364]+s2[onestepfnum+1-364]))}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Double Additive Seasonal Multiplicative Trend Exponential Smoothing With Autocorrelation Adjustment

```
doubleasmtauto<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="additive",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="additive",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])-(s1[1]+s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])-(a[1]+s2[1]))+(1-lambda3)*s1[1]
onestepf[1]<-(a[1]*b[1]^1)+s1[2]+s2[2]+lambda4^1*( as.vector(data)[initdatayear*364+1]-
((a[1])+s1[2]+s2[2]))
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])-(s1[m+1]+s2[m+1]))+(1-
lambda1)*(a[m]*b[m])
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*b[m]
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])-(a[m+1]+s2[m+1]))+(1-
lambda3)*s1[m+1]
onestepf[m+1]<-(a[m+1]*b[m+1]^1)+s1[m+2]+s2[m+2]+lambda4*(as.vector(data)[initdatayear*364+1+m]-
((a[m+1])+s1[m+2]+s2[m+2]))}
if(onestepfnum>363){
```

```
init2<-as.vector(data)[1:((initdatayear+1)*364)]
ts22<-ts(init2,start=1,frequency=7)
d22<-decompose(ts22,type="additive",filter=NULL)
s22<-(as.vector(d22$seasonal))[1:364]
s2[365:728]<-s22
        for(n in 363:(onestepfnum-1)){
a[n+1]<-lambda1*((as.vector(data)[initdatayear*364+1+n])-(s1[n+1]+s2[n+1]))+(1-lambda1)*(a[n]*b[n])
b[n+1]<-lambda2*(a[n+1]/a[n])+(1-lambda2)*b[n]
s1[364+n+1]<-lambda3*((as.vector(data)[initdatayear*364+1+n])-(a[n+1]+s2[n+1]))+(1-lambda3)*s1[n+1]
onestepf[n+1]<-(a[n+1]*b[n+1]^1)+s1[n+2]+s2[n+2]+lambda4*(as.vector(data)[initdatayear*364+1+n]-
((a[n+1])+s1[n+2]+s2[n+2]))}
}
if(onestepfnum>727){
init3<-as.vector(data)[1:((initdatayear+2)*364)]
ts23<-ts(init3,start=1,frequency=7)
d23<-decompose(ts23,type="additive",filter=NULL)
s23<-(as.vector(d23$seasonal))[1:364]
s2[729:1092]<-s23
        for(p in 727:(onestepfnum-1)){
a[p+1]<-lambda1*((as.vector(data)[initdatayear*364+1+p])-(s1[p+1]+s2[p+1]))+(1-lambda1)*(a[p]*b[p])
b[p+1]<-lambda2*(a[p+1]/a[p])+(1-lambda2)*b[p]
s1[364+p+1]<-lambda3*((as.vector(data)[initdatayear*364+1+p])-(a[p+1]+s2[p+1]))+(1-lambda3)*s1[p+1]
onestepf[p+1]<-(a[p+1]*b[p+1]^1)+s1[p+2]+s2[p+2]+lambda4*(as.vector(data)[initdatayear*364+1+p]-
((a[p+1])+s1[p+2]+s2[p+2]))}
}
if(onestepfnum>1091){
init4<-as.vector(data)[1:((initdatayear+3)*364)]
ts24<-ts(init4,start=1,frequency=7)
d24<-decompose(ts24,type="additive",filter=NULL)
s24<-(as.vector(d24$seasonal))[1:364]
s2[1093:1456]<-s24
        for(k in 1091:(onestepfnum-1)){
a[k+1]<-lambda1*((as.vector(data)[initdatayear*364+1+k])-(s1[k+1]+s2[k+1]))+(1-lambda1)*(a[k]*b[k])
b[k+1]<-lambda2*(a[k+1]/a[k])+(1-lambda2)*b[k]
s1[364+k+1]<-lambda3*((as.vector(data)[initdatayear*364+1+k])-(a[k+1]+s2[k+1]))+(1-lambda3)*s1[k+1]
onestepf[k+1]<-(a[k+1]*b[k+1]^1)+s1[k+2]+s2[k+2]+lambda4*(as.vector(data)[initdatayear*364+1+k]-
((a[k+1])+s1[k+2]+s2[k+2]))}
}
if(onestepfnum>1455){
init5<-as.vector(data)[1:((initdatayear+4)*364)]
```

```
ts25<-ts(init5,start=1,frequency=7)
d25<-decompose(ts25,type="additive",filter=NULL)
s25<-(as.vector(d25$seasonal))[1:364]
s2[1457:1820]<-s25
        for(r in 1455:(onestepfnum-1)){
a[r+1]<-lambda1*((as.vector(data)[initdatayear*364+1+r])-(s1[r+1]+s2[r+1]))+(1-lambda1)*(a[r]*b[r])
b[r+1]<-lambda2*(a[r+1]/a[r])+(1-lambda2)*b[r]
s1[364+r+1]<-lambda3*((as.vector(data)[initdatayear*364+1+r])-(a[r+1]+s2[r+1]))+(1-lambda3)*s1[r+1]
onestepf[r+1]<-(a[r+1]*b[r+1]^1)+s1[r+2]+s2[r+2]+lambda4*(as.vector(data)[initdatayear*364+1+r]-
((a[r+1])+s1[r+2]+s2[r+2]))}
}
if(onestepfnum>1819){
init6<-as.vector(data)[1:((initdatayear+5)*364)]
ts26<-ts(init6,start=1,frequency=7)
d26<-decompose(ts26,type="additive",filter=NULL)
s26<-(as.vector(d26$seasonal))[1:364]
s2[1821:2184]<-s26
        for(j in 1819:(onestepfnum-1)){
a[j+1]<-lambda1*((as.vector(data)[initdatayear*364+1+j])-(s1[j+1]+s2[j+1]))+(1-lambda1)*(a[j]*b[j])
b[j+1]<-lambda2*(a[j+1]/a[j])+(1-lambda2)*b[j]
s1[364+j+1]<-lambda3*((as.vector(data)[initdatayear*364+1+j])-(a[j+1]+s2[j+1]))+(1-lambda3)*s1[j+1]
onestepf[j+1]<-(a[j+1]*b[j+1]^1)+s1[j+2]+s2[j+2]+lambda4*(as.vector(data)[initdatayear*364+1+j]-
((a[j+1])+s1[j+2]+s2[j+2]))}
}
if(onestepfnum>2183){
init7<-as.vector(data)[1:((initdatayear+6)*364)]
ts27<-ts(init7,start=1,frequency=7)
d27<-decompose(ts27,type="additive",filter=NULL)
s27<-(as.vector(d27$seasonal))[1:364]
s2[2185:2548]<-s27
        for(y in 2183:(onestepfnum-1)){
a[y+1]<-lambda1*((as.vector(data)[initdatayear*364+1+y])-(s1[y+1]+s2[y+1]))+(1-lambda1)*(a[y]*b[y])
b[y+1]<-lambda2*(a[y+1]/a[y])+(1-lambda2)*b[y]
s1[364+y+1]<-lambda3*((as.vector(data)[initdatayear*364+1+y])-(a[y+1]+s2[y+1]))+(1-lambda3)*s1[y+1]
onestepf[y+1]<-(a[y+1]*b[y+1]^1)+s1[y+2]+s2[y+2]+lambda4*(as.vector(data)[initdatayear*364+1+y]-
((a[y+1])+s1[y+2]+s2[y+2]))}
}
if(onestepfnum>2547){
init8<-as.vector(data)[1:((initdatayear+7)*364)]
ts28<-ts(init8,start=1,frequency=7)
```

```
d28<-decompose(ts28,type="additive",filter=NULL)
s28<-(as.vector(d28$seasonal))[1:364]
s2[2549:2912]<-s28
        for(o in 2547:(onestepfnum-1)){
a[o+1]<-lambda1*((as.vector(data)[initdatayear*364+1+o])-(s1[o+1]+s2[o+1]))+(1-lambda1)*(a[o]*b[o])
b[o+1]<-lambda2*(a[o+1]/a[o])+(1-lambda2)*b[o]
s1[364+o+1]<-lambda3*((as.vector(data)[initdatayear*364+1+o])-(a[o+1]+s2[o+1]))+(1-lambda3)*s1[o+1]
onestepf[o+1]<-(a[o+1]*b[o+1]^1)+s1[o+2]+s2[o+2]+lambda4*(as.vector(data)[initdatayear*364+1+o]-
((a[o+1])+s1[o+2]+s2[o+2]))}
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
onestepf[i+onestepfnum]<-(a[onestepfnum]*b[onestepfnum]^(i+1))+s1[i+onestepfnum+1-
364]+s2[i+onestepfnum+1-364]+lambda4^(i)*(as.vector(data)[initdatayear*364+1+onestepfnum]-
((a[onestepfnum-1]*b[onestepfnum-1])+s1[onestepfnum+1-364]+s2[onestepfnum+1-364]))}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Double Multiplicative Seasonal Additive Trend Exponential Smoothing

```
doublemsat<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="multiplicative",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="multiplicative",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
```

```
onestepf<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])/(s1[1]*s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
b[1]<-lambda2*(a[1]-(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*initbeta1
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])/(a[1]*s2[1]))+(1-lambda3)*s1[1]
onestepf[1]<-(a[1]+b[1]*1)*s1[2]*s2[2]
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])/(s1[m+1]*s2[m+1]))+(1-
lambda1)*(a[m]+b[m])
b[m+1]<-lambda2*(a[m+1]-a[m])+(1-lambda2)*b[m]
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s2[m+1]))+(1-
lambda3)*s1[m+1]
onestepf[m+1]<-(a[m+1]+b[m+1]*1)*s1[m+2]*s2[m+2]}
if(onestepfnum>363){
init2<-as.vector(data)[1:((initdatayear+1)*364)]
ts22<-ts(init2,start=1,frequency=7)
d22<-decompose(ts22,type="multiplicative",filter=NULL)
s22<-(as.vector(d22$seasonal))[1:364]
s2[365:728]<-s22
        for(n in 363:(onestepfnum-1)){
a[n+1]<-lambda1*((as.vector(data)[initdatayear*364+1+n])/(s1[n+1]*s2[n+1]))+(1-lambda1)*(a[n]+b[n])
b[n+1]<-lambda2*(a[n+1]-a[n])+(1-lambda2)*b[n]
s1[364+n+1]<-lambda3*((as.vector(data)[initdatayear*364+1+n])/(a[n+1]*s2[n+1]))+(1-lambda3)*s1[n+1]
onestepf[n+1]<-(a[n+1]+b[n+1]*1)*s1[n+2]*s2[n+2]}
}
if(onestepfnum>727){
init3<-as.vector(data)[1:((initdatayear+2)*364)]
ts23<-ts(init3,start=1,frequency=7)
d23<-decompose(ts23,type="multiplicative",filter=NULL)
s23<-(as.vector(d23$seasonal))[1:364]
s2[729:1092]<-s23
        for(p in 727:(onestepfnum-1)){
a[p+1]<-lambda1*((as.vector(data)[initdatayear*364+1+p])/(s1[p+1]*s2[p+1]))+(1-lambda1)*(a[p]+b[p])
b[p+1]<-lambda2*(a[p+1]-a[p])+(1-lambda2)*b[p]
s1[364+p+1]<-lambda3*((as.vector(data)[initdatayear*364+1+p])/(a[p+1]*s2[p+1]))+(1-lambda3)*s1[p+1]
onestepf[p+1]<-(a[p+1]+b[p+1]*1)*s1[p+2]*s2[p+2]}
}
if(onestepfnum>1091){
```

```
init4<-as.vector(data)[1:((initdatayear+3)*364)]
ts24<-ts(init4,start=1,frequency=7)
d24<-decompose(ts24,type="multiplicative",filter=NULL)
s24<-(as.vector(d24$seasonal))[1:364]
s2[1093:1456]<-s24
        for(k in 1091:(onestepfnum-1)){
a[k+1]<-lambda1*((as.vector(data)[initdatayear*364+1+k])/(s1[k+1]*s2[k+1]))+(1-lambda1)*(a[k]+b[k])
b[k+1]<-lambda2*(a[k+1]-a[k])+(1-lambda2)*b[k]
s1[364+k+1]<-lambda3*((as.vector(data)[initdatayear*364+1+k])/(a[k+1]*s2[k+1]))+(1-lambda3)*s1[k+1]
onestepf[k+1]<-(a[k+1]+b[k+1]*1)*s1[k+2]*s2[k+2]}
}
if(onestepfnum>1455){
init5<-as.vector(data)[1:((initdatayear+4)*364)]
ts25<-ts(init5,start=1,frequency=7)
d25<-decompose(ts25,type="multiplicative",filter=NULL)
s25<-(as.vector(d25$seasonal))[1:364]
s2[1457:1820]<-s25
        for(r in 1455:(onestepfnum-1)){
a[r+1]<-lambda1*((as.vector(data)[initdatayear*364+1+r])/(s1[r+1]*s2[r+1]))+(1-lambda1)*(a[r]+b[r])
b[r+1]<-lambda2*(a[r+1]-a[r])+(1-lambda2)*b[r]
s1[364+r+1]<-lambda3*((as.vector(data)[initdatayear*364+1+r])/(a[r+1]*s2[r+1]))+(1-lambda3)*s1[r+1]
onestepf[r+1]<-(a[r+1]+b[r+1]*1)*s1[r+2]*s2[r+2]}
}
if(onestepfnum>1819){
init6<-as.vector(data)[1:((initdatayear+5)*364)]
ts26<-ts(init6,start=1,frequency=7)
d26<-decompose(ts26,type="multiplicative",filter=NULL)
s26<-(as.vector(d26$seasonal))[1:364]
s2[1821:2184]<-s26
        for(j in 1819:(onestepfnum-1)){
a[j+1]<-lambda1*((as.vector(data)[initdatayear*364+1+j])/(s1[j+1]*s2[j+1]))+(1-lambda1)*(a[j]+b[j])
b[j+1]<-lambda2*(a[j+1]-a[j])+(1-lambda2)*b[j]
s1[364+j+1]<-lambda3*((as.vector(data)[initdatayear*364+1+j])/(a[j+1]*s2[j+1]))+(1-lambda3)*s1[j+1]
onestepf[j+1]<-(a[j+1]+b[j+1]*1)*s1[j+2]*s2[j+2]}
}
if(onestepfnum>2183){
init7<-as.vector(data)[1:((initdatayear+6)*364)]
ts27<-ts(init7,start=1,frequency=7)
d27<-decompose(ts27,type="multiplicative",filter=NULL)
s27<-(as.vector(d27$seasonal))[1:364]
```

```
s2[2185:2548]<-s27
        for(y in 2183:(onestepfnum-1)){
a[y+1]<-lambda1*((as.vector(data)[initdatayear*364+1+y])/(s1[y+1]*s2[y+1]))+(1-lambda1)*(a[y]+b[y])
b[y+1]<-lambda2*(a[y+1]-a[y])+(1-lambda2)*b[y]
s1[364+y+1]<-lambda3*((as.vector(data)[initdatayear*364+1+y])/(a[y+1]*s2[y+1]))+(1-lambda3)*s1[y+1]
onestepf[y+1]<-(a[y+1]+b[y+1]*1)*s1[y+2]*s2[y+2]}
}
if(onestepfnum>2547){
init8<-as.vector(data)[1:((initdatayear+7)*364)]
ts28<-ts(init8,start=1,frequency=7)
d28<-decompose(ts28,type="multiplicative",filter=NULL)
s28<-(as.vector(d28$seasonal))[1:364]
s2[2549:2912]<-s28
        for(o in 2547:(onestepfnum-1)){
a[o+1]<-lambda1*((as.vector(data)[initdatayear*364+1+o])/(s1[o+1]*s2[o+1]))+(1-lambda1)*(a[o]+b[o])
b[o+1]<-lambda2*(a[o+1]-a[o])+(1-lambda2)*b[o]
s1[364+o+1]<-lambda3*((as.vector(data)[initdatayear*364+1+o])/(a[o+1]*s2[o+1]))+(1-lambda3)*s1[o+1]
onestepf[o+1]<-(a[o+1]+b[o+1]*1)*s1[o+2]*s2[o+2]}
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
onestepf[i+onestepfnum]<-(a[onestepfnum]+b[onestepfnum]*(i+1))*s1[i+onestepfnum+1-
364]*s2[i+onestepfnum+1-364]}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Double Multiplicative Seasonal Additive Trend Exponential Smoothing With Autocorrelation Adjustment

```
doublemsatauto<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
```

```
d1<-decompose(ts1,type="multiplicative",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="multiplicative",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])/(s1[1]*s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
b[1]<-lambda2*(a[1]-(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*initbeta1
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])/(a[1]*s2[1]))+(1-lambda3)*s1[1]
onestepf[1]<-(a[1]+b[1]*1)*s1[2]*s2[2]+lambda4^1*(as.vector(data)[initdatayear*364+1]-
((a[1])*s1[2]*s2[2]))
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])/(s1[m+1]*s2[m+1]))+(1-
lambda1)*(a[m]+b[m])
b[m+1]<-lambda2*(a[m+1]-a[m])+(1-lambda2)*b[m]
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s2[m+1]))+(1-
lambda3)*s1[m+1]
onestepf[m+1]<-(a[m+1]+b[m+1]*1)*s1[m+2]*s2[m+2]+lambda4*(as.vector(data)[initdatayear*364+1+m]-
((a[m+1])*s1[m+2]*s2[m+2]))}
if(onestepfnum>363){
init2<-as.vector(data)[1:((initdatayear+1)*364)]
ts22<-ts(init2,start=1,frequency=7)
d22<-decompose(ts22,type="multiplicative",filter=NULL)
s22<-(as.vector(d22$seasonal))[1:364]
s2[365:728]<-s22
        for(n in 363:(onestepfnum-1)){
a[n+1]<-lambda1*((as.vector(data)[initdatayear*364+n+1])/(s1[n+1]*s2[n+1]))+(1-lambda1)*(a[n]+b[n])
b[n+1]<-lambda2*(a[n+1]-a[n])+(1-lambda2)*b[n]
s1[364+n+1]<-lambda3*((as.vector(data)[initdatayear*364+n+1])/(a[n+1]*s2[n+1]))+(1-lambda3)*s1[n+1]
onestepf[n+1]<-(a[n+1]+b[n+1]*1)*s1[n+2]*s2[n+2]+lambda4*(as.vector(data)[initdatayear*364+n+1]-
((a[n+1])*s1[n+2]*s2[n+2]))}
}
```

```
if(onestepfnum>727){
init3<-as.vector(data)[1:((initdatayear+2)*364)]
ts23<-ts(init3,start=1,frequency=7)
d23<-decompose(ts23,type="multiplicative",filter=NULL)
s23<-(as.vector(d23$seasonal))[1:364]
s2[729:1092]<-s23
        for(p in 727:(onestepfnum-1)){
a[p+1]<-lambda1*((as.vector(data)[initdatayear*364+p+1])/(s1[p+1]*s2[p+1]))+(1-lambda1)*(a[p]+b[p])
b[p+1]<-lambda2*(a[p+1]-a[p])+(1-lambda2)*b[p]
s1[364+p+1]<-lambda3*((as.vector(data)[initdatayear*364+p+1])/(a[p+1]*s2[p+1]))+(1-lambda3)*s1[p+1]
onestepf[p+1]<-(a[p+1]+b[p+1]*1)*s1[p+2]*s2[p+2]+lambda4*(as.vector(data)[initdatayear*364+p+1]-
((a[p+1])*s1[p+2]*s2[p+2]))}
}
if(onestepfnum>1091){
init4<-as.vector(data)[1:((initdatayear+3)*364)]
ts24<-ts(init4,start=1,frequency=7)
d24<-decompose(ts24,type="multiplicative",filter=NULL)
s24<-(as.vector(d24$seasonal))[1:364]
s2[1093:1456]<-s24
        for(k in 1091:(onestepfnum-1)){
a[k+1]<-lambda1*((as.vector(data)[initdatayear*364+k+1])/(s1[k+1]*s2[k+1]))+(1-lambda1)*(a[k]+b[k])
b[k+1]<-lambda2*(a[k+1]-a[k])+(1-lambda2)*b[k]
s1[364+k+1]<-lambda3*((as.vector(data)[initdatayear*364+k+1])/(a[k+1]*s2[k+1]))+(1-lambda3)*s1[k+1]
onestepf[k+1]<-(a[k+1]+b[k+1]*1)*s1[k+2]*s2[k+2]+lambda4*(as.vector(data)[initdatayear*364+k+1]-
((a[k+1])*s1[k+2]*s2[k+2]))}
}
if(onestepfnum>1455){
init5<-as.vector(data)[1:((initdatayear+4)*364)]
ts25<-ts(init5,start=1,frequency=7)
d25<-decompose(ts25,type="multiplicative",filter=NULL)
s25<-(as.vector(d25$seasonal))[1:364]
s2[1457:1820]<-s25
        for(r in 1455:(onestepfnum-1)){
a[r+1]<-lambda1*((as.vector(data)[initdatayear*364+r+1])/(s1[r+1]*s2[r+1]))+(1-lambda1)*(a[r]+b[r])
b[r+1]<-lambda2*(a[r+1]-a[r])+(1-lambda2)*b[r]
s1[364+r+1]<-lambda3*((as.vector(data)[initdatayear*364+r+1])/(a[r+1]*s2[r+1]))+(1-lambda3)*s1[r+1]
onestepf[r+1]<-(a[r+1]+b[r+1]*1)*s1[r+2]*s2[r+2]+lambda4*(as.vector(data)[initdatayear*364+r+1]-
((a[r+1])*s1[r+2]*s2[r+2]))}
}
if(onestepfnum>1819){
```

```
init6<-as.vector(data)[1:((initdatayear+5)*364)]
ts26<-ts(init6,start=1,frequency=7)
d26<-decompose(ts26,type="multiplicative",filter=NULL)
s26<-(as.vector(d26$seasonal))[1:364]
s2[1821:2184]<-s26
        for(j in 1819:(onestepfnum-1)){
a[j+1]<-lambda1*((as.vector(data)[initdatayear*364+j+1])/(s1[j+1]*s2[j+1]))+(1-lambda1)*(a[j]+b[j])
b[j+1]<-lambda2*(a[j+1]-a[j])+(1-lambda2)*b[j]
s1[364+j+1]<-lambda3*((as.vector(data)[initdatayear*364+j+1])/(a[j+1]*s2[j+1]))+(1-lambda3)*s1[j+1]
onestepf[j+1]<-(a[j+1]+b[j+1]*1)*s1[j+2]*s2[j+2]+lambda4*(as.vector(data)[initdatayear*364+j+1]-
((a[j+1])*s1[j+2]*s2[j+2]))}
}
if(onestepfnum>2183){
init7<-as.vector(data)[1:((initdatayear+6)*364)]
ts27<-ts(init7,start=1,frequency=7)
d27<-decompose(ts27,type="multiplicative",filter=NULL)
s27<-(as.vector(d27$seasonal))[1:364]
s2[2185:2548]<-s27
        for(y in 2183:(onestepfnum-1)){
a[y+1]<-lambda1*((as.vector(data)[initdatayear*364+y+1])/(s1[y+1]*s2[y+1]))+(1-lambda1)*(a[y]+b[y])
b[y+1]<-lambda2*(a[y+1]-a[y])+(1-lambda2)*b[y]
s1[364+y+1]<-lambda3*((as.vector(data)[initdatayear*364+y+1])/(a[y+1]*s2[y+1]))+(1-lambda3)*s1[y+1]
onestepf[y+1]<-(a[y+1]+b[y+1]*1)*s1[y+2]*s2[y+2]+lambda4*(as.vector(data)[initdatayear*364+y+1]-
((a[y+1])*s1[y+2]*s2[y+2]))}
}
if(onestepfnum>2547){
init8<-as.vector(data)[1:((initdatayear+7)*364)]
ts28<-ts(init8,start=1,frequency=7)
d28<-decompose(ts28,type="multiplicative",filter=NULL)
s28<-(as.vector(d28$seasonal))[1:364]
s2[2549:2912]<-s28
        for(o in 2547:(onestepfnum-1)){
a[o+1]<-lambda1*((as.vector(data)[initdatayear*364+o+1])/(s1[o+1]*s2[o+1]))+(1-lambda1)*(a[o]+b[o])
b[o+1]<-lambda2*(a[o+1]-a[o])+(1-lambda2)*b[o]
s1[364+o+1]<-lambda3*((as.vector(data)[initdatayear*364+o+1])/(a[o+1]*s2[o+1]))+(1-lambda3)*s1[o+1]
onestepf[o+1]<-(a[o+1]+b[o+1]*1)*s1[o+2]*s2[o+2]+lambda4*(as.vector(data)[initdatayear*364+o+1]-
((a[o+1])*s1[o+2]*s2[o+2]))}
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
```

```
else{
        for(i in 1:n.ahead){
onestepf[i+onestepfnum]<-(a[onestepfnum]+b[onestepfnum]*(i+1))*s1[i+onestepfnum+1-
364]*s2[i+onestepfnum+1-364]+lambda4^(i)*(as.vector(data)[initdatayear*364+onestepfnum+1]-
((a[onestepfnum-1]+b[onestepfnum-1])*s1[onestepfnum+1-364]*s2[onestepfnum+1-364]))}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Double Multiplicative Seasonal Multiplicative Damped Trend Exponential Smoothing

```
doublemsdmt<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="multiplicative",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="multiplicative",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
e<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])/(s1[1]*s2[1]))+(1-
lambda1)*((initbeta0+initbeta1*initdatayear*364)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda4)
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
(((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda4)
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])/(a[1]*s2[1]))+(1-lambda3)*s1[1]
onestepf[1]<-(a[1]*b[1]^lambda4)*s1[2]*s2[2]
```

```
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])/(s1[m+1]*s2[m+1]))+(1-
lambda1)*(a[m]*b[m]^lambda4)
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*(b[m]^lambda4)
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s2[m+1]))+(1-
lambda3)*s1[m+1]
onestepf[m+1]<-(a[m+1]*b[m+1]^lambda4)*s1[m+2]*s2[m+2]}
if(onestepfnum>363){
init2<-as.vector(data)[1:((initdatayear+1)*364)]
ts22<-ts(init2,start=1,frequency=7)
d22<-decompose(ts22,type="multiplicative",filter=NULL)
s22<-(as.vector(d22$seasonal))[1:364]
s2[365:728]<-s22
        for(n in 363:(onestepfnum-1)){
a[n+1]<-lambda1*((as.vector(data)[initdatayear*364+1+n])/(s1[n+1]*s2[n+1]))+(1-
lambda1)*(a[n]*(b[n]^lambda4))
b[n+1]<-lambda2*(a[n+1]/a[n])+(1-lambda2)*(b[n]^lambda4)
s1[364+n+1]<-lambda3*((as.vector(data)[initdatayear*364+1+n])/(a[n+1]*s2[n+1]))+(1-lambda3)*s1[n+1]
onestepf[n+1]<-(a[n+1]*b[n+1]^lambda4)*s1[n+2]*s2[n+2]}
}
if(onestepfnum>727){
init3<-as.vector(data)[1:((initdatayear+2)*364)]
ts23<-ts(init3,start=1,frequency=7)
d23<-decompose(ts23,type="multiplicative",filter=NULL)
s23<-(as.vector(d23$seasonal))[1:364]
s2[729:1092]<-s23
        for(p in 727:(onestepfnum-1)){
a[p+1]<-lambda1*((as.vector(data)[initdatayear*364+1+p])/(s1[p+1]*s2[p+1]))+(1-
lambda1)*(a[p]*(b[p]^lambda4))
b[p+1]<-lambda2*(a[p+1]/a[p])+(1-lambda2)*(b[p]^lambda4)
s1[364+p+1]<-lambda3*((as.vector(data)[initdatayear*364+1+p])/(a[p+1]*s2[p+1]))+(1-lambda3)*s1[p+1]
onestepf[p+1]<-(a[p+1]*b[p+1]^lambda4)*s1[p+2]*s2[p+2]}
}
if(onestepfnum>1091){
init4<-as.vector(data)[1:((initdatayear+3)*364)]
ts24<-ts(init4,start=1,frequency=7)
d24<-decompose(ts24,type="multiplicative",filter=NULL)
s24<-(as.vector(d24$seasonal))[1:364]
s2[1093:1456]<-s24
        for(k in 1091:(onestepfnum-1)){
```

```
a[k+1]<-lambda1*((as.vector(data)[initdatayear*364+1+k])/(s1[k+1]*s2[k+1]))+(1-
lambda1)*(a[k]*b[k]^lambda4)
b[k+1]<-lambda2*(a[k+1]/a[k])+(1-lambda2)*b[k]^lambda4
s1[364+k+1]<-lambda3*((as.vector(data)[initdatayear*364+1+k])/(a[k+1]*s2[k+1]))+(1-lambda3)*s1[k+1]
onestepf[k+1]<-(a[k+1]*b[k+1]^lambda4)*s1[k+2]*s2[k+2]}
}
if(onestepfnum>1455){
init5<-as.vector(data)[1:((initdatayear+4)*364)]
ts25<-ts(init5,start=1,frequency=7)
d25<-decompose(ts25,type="multiplicative",filter=NULL)
s25<-(as.vector(d25$seasonal))[1:364]
s2[1457:1820]<-s25
        for(r in 1455:(onestepfnum-1)){
a[r+1]<-lambda1*((as.vector(data)[initdatayear*364+1+r])/(s1[r+1]*s2[r+1]))+(1-
lambda1)*(a[r]*b[r]^lambda4)
b[r+1]<-lambda2*(a[r+1]/a[r])+(1-lambda2)*b[r]^lambda4
s1[364+r+1]<-lambda3*((as.vector(data)[initdatayear*364+1+r])/(a[r+1]*s2[r+1]))+(1-lambda3)*s1[r+1]
onestepf[r+1]<-(a[r+1]*b[r+1]^lambda4)*s1[r+2]*s2[r+2]}
}
if(onestepfnum>1819){
init6<-as.vector(data)[1:((initdatayear+5)*364)]
ts26<-ts(init6,start=1,frequency=7)
d26<-decompose(ts26,type="multiplicative",filter=NULL)
s26<-(as.vector(d26$seasonal))[1:364]
s2[1821:2184]<-s26
        for(j in 1819:(onestepfnum-1)){
a[j+1]<-lambda1*((as.vector(data)[initdatayear*364+1+j])/(s1[j+1]*s2[j+1]))+(1-
lambda1)*(a[j]*b[j]^lambda4)
b[j+1]<-lambda2*(a[j+1]/a[j])+(1-lambda2)*b[j]^lambda4
s1[364+j+1]<-lambda3*((as.vector(data)[initdatayear*364+1+j])/(a[j+1]*s2[j+1]))+(1-lambda3)*s1[j+1]
onestepf[j+1]<-(a[j+1]*b[j+1]^lambda4)*s1[j+2]*s2[j+2]}
}
if(onestepfnum>2183){
init7<-as.vector(data)[1:((initdatayear+6)*364)]
ts27<-ts(init7,start=1,frequency=7)
d27<-decompose(ts27,type="multiplicative",filter=NULL)
s27<-(as.vector(d27$seasonal))[1:364]
s2[2185:2548]<-s27
        for(y in 2183:(onestepfnum-1)){
```

```
a[y+1]<-lambda1*((as.vector(data)[initdatayear*364+1+y])/(s1[y+1]*s2[y+1]))+(1-
lambda1)*(a[y]*b[y]^lambda4)
b[y+1]<-lambda2*(a[y+1]/a[y])+(1-lambda2)*b[y]^lambda4
s1[364+y+1]<-lambda3*((as.vector(data)[initdatayear*364+1+y])/(a[y+1]*s2[y+1]))+(1-lambda3)*s1[y+1]
onestepf[y+1]<-(a[y+1]*b[y+1]^lambda4)*s1[y+2]*s2[y+2]}
}
if(onestepfnum>2547){
init8<-as.vector(data)[1:((initdatayear+7)*364)]
ts28<-ts(init8,start=1,frequency=7)
d28<-decompose(ts28,type="multiplicative",filter=NULL)
s28<-(as.vector(d28$seasonal))[1:364]
s2[2549:2912]<-s28
        for(o in 2547:(onestepfnum-1)){
a[o+1]<-lambda1*((as.vector(data)[initdatayear*364+1+o])/(s1[o+1]*s2[o+1]))+(1-
lambda1)*(a[o]*b[o]^lambda4)
b[o+1]<-lambda2*(a[o+1]/a[o])+(1-lambda2)*b[o]^lambda4
s1[364+o+1]<-lambda3*((as.vector(data)[initdatayear*364+1+o])/(a[o+1]*s2[o+1]))+(1-lambda3)*s1[o+1]
onestepf[o+1]<-(a[o+1]*b[o+1]^lambda4)*s1[o+2]*s2[o+2]}
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
        for(z in 1:(i+1)){
e[z]<-(lambda4^(z))}
se<-sum(e)
onestepf[i+onestepfnum]<-(a[onestepfnum]*b[onestepfnum]^(se))*s1[i+onestepfnum+1-
364]*s2[i+onestepfnum+1-364]}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Double Multiplicative Seasonal Multiplicative Damped Trend Exponential Smoothing With Autocorrelation Adjustment

```
doublemsdmtauto<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
```

```
lambda5<-lambda[5]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="multiplicative",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="multiplicative",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
e<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])/(s1[1]*s2[1]))+(1-
lambda1)*((initbeta0+initbeta1*initdatayear*364)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda5)
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
(((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda5)
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])/(a[1]*s2[1]))+(1-lambda3)*s1[1]
onestepf[1]<-(a[1]*b[1]^lambda5)*s1[2]*s2[2]+lambda4^1*( as.vector(data)[initdatayear*364+1]-((a[1])*
s1[2]*s2[2]))
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])/(s1[m+1]*s2[m+1]))+(1-
lambda1)*(a[m]*b[m]^lambda5)
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*(b[m]^lambda5)
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s2[m+1]))+(1-
lambda3)*s1[m+1]
onestepf[m+1]<-
(a[m+1]*b[m+1]^lambda5)*s1[m+2]*s2[m+2]+lambda4*(as.vector(data)[initdatayear*364+1+m]-
((a[m+1])*s1[m+2]*s2[m+2]))}
if(onestepfnum>363){
init2<-as.vector(data)[1:((initdatayear+1)*364)]
ts22<-ts(init2,start=1,frequency=7)
d22<-decompose(ts22,type="multiplicative",filter=NULL)
s22<-(as.vector(d22$seasonal))[1:364]
```

```
s2[365:728]<-s22
        for(n in 363:(onestepfnum-1)){
a[n+1]<-lambda1*((as.vector(data)[initdatayear*364+1+n])/(s1[n+1]*s2[n+1]))+(1-
lambda1)*(a[n]*b[n]^lambda5)
b[n+1]<-lambda2*(a[n+1]/a[n])+(1-lambda2)*(b[n]^lambda5)
s1[364+n+1]<-lambda3*((as.vector(data)[initdatayear*364+1+n])/(a[n+1]*s2[n+1]))+(1-lambda3)*s1[n+1]
onestepf[n+1]<-
(a[n+1]*b[n+1]^lambda5)*s1[n+2]*s2[n+2]+lambda4*(as.vector(data)[initdatayear*364+1+n]-
((a[n+1])*s1[n+2]*s2[n+2]))}
}
if(onestepfnum>727){
init3<-as.vector(data)[1:((initdatayear+2)*364)]
ts23<-ts(init3,start=1,frequency=7)
d23<-decompose(ts23,type="multiplicative",filter=NULL)
s23<-(as.vector(d23$seasonal))[1:364]
s2[729:1092]<-s23
        for(p in 727:(onestepfnum-1)){
a[p+1]<-lambda1*((as.vector(data)[initdatayear*364+1+p])/(s1[p+1]*s2[p+1]))+(1-
lambda1)*(a[p]*b[p]^lambda5)
b[p+1]<-lambda2*(a[p+1]/a[p])+(1-lambda2)*(b[p]^lambda5)
s1[364+p+1]<-lambda3*((as.vector(data)[initdatayear*364+1+p])/(a[p+1]*s2[p+1]))+(1-lambda3)*s1[p+1]
onestepf[p+1]<-
(a[p+1]*b[p+1]^lambda5)*s1[p+2]*s2[p+2]+lambda4*(as.vector(data)[initdatayear*364+1+p]-
((a[p+1])*s1[p+2]*s2[p+2]))}
}
if(onestepfnum>1091){
init4<-as.vector(data)[1:((initdatayear+3)*364)]
ts24<-ts(init4,start=1,frequency=7)
d24<-decompose(ts24,type="multiplicative",filter=NULL)
s24<-(as.vector(d24$seasonal))[1:364]
s2[1093:1456]<-s24
        for(k in 1091:(onestepfnum-1)){
a[k+1]<-lambda1*((as.vector(data)[initdatayear*364+1+k])/(s1[k+1]*s2[k+1]))+(1-
lambda1)*(a[k]*b[k]^lambda5)
b[k+1]<-lambda2*(a[k+1]/a[k])+(1-lambda2)*(b[k]^lambda5)
s1[364+k+1]<-lambda3*((as.vector(data)[initdatayear*364+1+k])/(a[k+1]*s2[k+1]))+(1-lambda3)*s1[k+1]
onestepf[k+1]<-
(a[k+1]*b[k+1]^lambda5)*s1[k+2]*s2[k+2]+lambda4*(as.vector(data)[initdatayear*364+1+k]-
((a[k+1])*s1[k+2]*s2[k+2]))}
}
```

```
if(onestepfnum>1455){
init5<-as.vector(data)[1:((initdatayear+4)*364)]
ts25<-ts(init5,start=1,frequency=7)
d25<-decompose(ts25,type="multiplicative",filter=NULL)
s25<-(as.vector(d25$seasonal))[1:364]
s2[1457:1820]<-s25
        for(r in 1455:(onestepfnum-1)){
a[r+1]<-lambda1*((as.vector(data)[initdatayear*364+1+r])/(s1[r+1]*s2[r+1]))+(1-
lambda1)*(a[r]*b[r]^lambda5)
b[r+1]<-lambda2*(a[r+1]/a[r])+(1-lambda2)*(b[r]^lambda5)
s1[364+r+1]<-lambda3*((as.vector(data)[initdatayear*364+1+r])/(a[r+1]*s2[r+1]))+(1-lambda3)*s1[r+1]
onestepf[r+1]<-(a[r+1]*b[r+1]^lambda5)*s1[r+2]*s2[r+2]+lambda4*(as.vector(data)[initdatayear*364+1+r]-
((a[r+1])*s1[r+2]*s2[r+2]))}
}
if(onestepfnum>1819){
init6<-as.vector(data)[1:((initdatayear+5)*364)]
ts26<-ts(init6,start=1,frequency=7)
d26<-decompose(ts26,type="multiplicative",filter=NULL)
s26<-(as.vector(d26$seasonal))[1:364]
s2[1821:2184]<-s26
        for(j in 1819:(onestepfnum-1)){
a[j+1]<-lambda1*((as.vector(data)[initdatayear*364+1+j])/(s1[j+1]*s2[j+1]))+(1-
lambda1)*(a[j]*b[j]^lambda5)
b[j+1]<-lambda2*(a[j+1]/a[j])+(1-lambda2)*(b[j]^lambda5)
s1[364+j+1]<-lambda3*((as.vector(data)[initdatayear*364+1+j])/(a[j+1]*s2[j+1]))+(1-lambda3)*s1[j+1]
onestepf[j+1]<-(a[j+1]*b[j+1]^lambda5)*s1[j+2]*s2[j+2]+lambda4*(as.vector(data)[initdatayear*364+1+j]-
((a[j+1])*s1[j+2]*s2[j+2]))}
}
if(onestepfnum>2183){
init7<-as.vector(data)[1:((initdatayear+6)*364)]
ts27<-ts(init7,start=1,frequency=7)
d27<-decompose(ts27,type="multiplicative",filter=NULL)
s27<-(as.vector(d27$seasonal))[1:364]
s2[2185:2548]<-s27
        for(y in 2183:(onestepfnum-1)){
a[y+1]<-lambda1*((as.vector(data)[initdatayear*364+1+y])/(s1[y+1]*s2[y+1]))+(1-
lambda1)*(a[y]*b[y]^lambda5)
b[y+1]<-lambda2*(a[y+1]/a[y])+(1-lambda2)*(b[y]^lambda5)
s1[364+y+1]<-lambda3*((as.vector(data)[initdatayear*364+1+y])/(a[y+1]*s2[y+1]))+(1-lambda3)*s1[y+1]
```

```
onestepf[y+1]<-
(a[y+1]*b[y+1]^lambda5)*s1[y+2]*s2[y+2]+lambda4*(as.vector(data)[initdatayear*364+1+y]-
((a[y+1])*s1[y+2]*s2[y+2]))}
}
if(onestepfnum>2547){
init8<-as.vector(data)[1:((initdatayear+7)*364)]
ts28<-ts(init8,start=1,frequency=7)
d28<-decompose(ts28,type="multiplicative",filter=NULL)
s28<-(as.vector(d28$seasonal))[1:364]
s2[2549:2912]<-s28
        for(o in 2547:(onestepfnum-1)){
a[o+1]<-lambda1*((as.vector(data)[initdatayear*364+1+o])/(s1[o+1]*s2[o+1]))+(1-
lambda1)*(a[o]*b[o]^lambda5)
b[o+1]<-lambda2*(a[o+1]/a[o])+(1-lambda2)*(b[o]^lambda5)
s1[364+o+1]<-lambda3*((as.vector(data)[initdatayear*364+1+o])/(a[o+1]*s2[o+1]))+(1-lambda3)*s1[o+1]
onestepf[o+1]<-
(a[o+1]*b[o+1]^lambda5)*s1[o+2]*s2[o+2]+lambda4*(as.vector(data)[initdatayear*364+1+o]-
((a[o+1])*s1[o+2]*s2[o+2]))}
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
        for(z in 1:(i+1)){
e[z]<-(lambda5^(z))}
se<-sum(e)
onestepf[i+onestepfnum]<-(a[onestepfnum]*b[onestepfnum]^(se))*s1[i+onestepfnum+1-
364]*s2[i+onestepfnum+1-364]+lambda4^(i)*(as.vector(data)[onestepfnum+initdatayear*364+1]-
((a[onestepfnum-1]*b[onestepfnum-1])*s1[onestepfnum+1-364]*s2[onestepfnum+1-364]))}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Double Multiplicative Seasonal Multiplicative Trend Exponential Smoothing With Autocorrelation Adjustment

```
doublemsmtauto<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
```

```
lambda4<-lambda[4]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="multiplicative",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="multiplicative",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])/(s1[1]*s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])/(a[1]*s2[1]))+(1-lambda3)*s1[1]
onestepf[1]<-(a[1]*b[1]^1)*s1[2]*s2[2]+lambda4^1*( as.vector(data)[initdatayear*364+1]-
((a[1])*s1[2]*s2[2]))
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])/(s1[m+1]*s2[m+1]))+(1-
lambda1)*(a[m]*b[m])
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*b[m]
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s2[m+1]))+(1-
lambda3)*s1[m+1]
onestepf[m+1]<-(a[m+1]*b[m+1]^1)*s1[m+2]*s2[m+2]+lambda4*(as.vector(data)[initdatayear*364+1+m]-
((a[m+1])*s1[m+2]*s2[m+2]))}
if(onestepfnum>363){
init2<-as.vector(data)[1:((initdatayear+1)*364)]
ts22<-ts(init2,start=1,frequency=7)
d22<-decompose(ts22,type="multiplicative",filter=NULL)
s22<-(as.vector(d22$seasonal))[1:364]
s2[365:728]<-s22
        for(n in 363:(onestepfnum-1)){
a[n+1]<-lambda1*((as.vector(data)[initdatayear*364+1+n])/(s1[n+1]*s2[n+1]))+(1-lambda1)*(a[n]*b[n])
```

```
b[n+1]<-lambda2*(a[n+1]/a[n])+(1-lambda2)*b[n]
s1[364+n+1]<-lambda3*((as.vector(data)[initdatayear*364+1+n])/(a[n+1]*s2[n+1]))+(1-lambda3)*s1[n+1]
onestepf[n+1]<-(a[n+1]*b[n+1]^1)*s1[n+2]*s2[n+2]+lambda4*(as.vector(data)[initdatayear*364+1+n]-
((a[n+1])*s1[n+2]*s2[n+2]))}
}
if(onestepfnum>727){
init3<-as.vector(data)[1:((initdatayear+2)*364)]
ts23<-ts(init3,start=1,frequency=7)
d23<-decompose(ts23,type="multiplicative",filter=NULL)
s23<-(as.vector(d23$seasonal))[1:364]
s2[729:1092]<-s23
        for(p in 727:(onestepfnum-1)){
a[p+1]<-lambda1*((as.vector(data)[initdatayear*364+1+p])/(s1[p+1]*s2[p+1]))+(1-lambda1)*(a[p]*b[p])
b[p+1]<-lambda2*(a[p+1]/a[p])+(1-lambda2)*b[p]
s1[364+p+1]<-lambda3*((as.vector(data)[initdatayear*364+1+p])/(a[p+1]*s2[p+1]))+(1-lambda3)*s1[p+1]
onestepf[p+1]<-(a[p+1]*b[p+1]^1)*s1[p+2]*s2[p+2]+lambda4*(as.vector(data)[initdatayear*364+1+p]-
((a[p+1])*s1[p+2]*s2[p+2]))}
}
if(onestepfnum>1091){
init4<-as.vector(data)[1:((initdatayear+3)*364)]
ts24<-ts(init4,start=1,frequency=7)
d24<-decompose(ts24,type="multiplicative",filter=NULL)
s24<-(as.vector(d24$seasonal))[1:364]
s2[1093:1456]<-s24
        for(k in 1091:(onestepfnum-1)){
a[k+1]<-lambda1*((as.vector(data)[initdatayear*364+1+k])/(s1[k+1]*s2[k+1]))+(1-lambda1)*(a[k]*b[k])
b[k+1]<-lambda2*(a[k+1]/a[k])+(1-lambda2)*b[k]
s1[364+k+1]<-lambda3*((as.vector(data)[initdatayear*364+1+k])/(a[k+1]*s2[k+1]))+(1-lambda3)*s1[k+1]
onestepf[k+1]<-(a[k+1]*b[k+1]^1)*s1[k+2]*s2[k+2]+lambda4*(as.vector(data)[initdatayear*364+1+k]-
((a[k+1])*s1[k+2]*s2[k+2]))}
}
if(onestepfnum>1455){
init5<-as.vector(data)[1:((initdatayear+4)*364)]
ts25<-ts(init5,start=1,frequency=7)
d25<-decompose(ts25,type="multiplicative",filter=NULL)
s25<-(as.vector(d25$seasonal))[1:364]
s2[1457:1820]<-s25
        for(r in 1455:(onestepfnum-1)){
a[r+1]<-lambda1*((as.vector(data)[initdatayear*364+1+r])/(s1[r+1]*s2[r+1]))+(1-lambda1)*(a[r]*b[r])
b[r+1]<-lambda2*(a[r+1]/a[r])+(1-lambda2)*b[r]
```

```
s1[364+r+1]<-lambda3*((as.vector(data)[initdatayear*364+1+r])/(a[r+1]*s2[r+1]))+(1-lambda3)*s1[r+1]
onestepf[r+1]<-(a[r+1]*b[r+1]^1)*s1[r+2]*s2[r+2]+lambda4*(as.vector(data)[initdatayear*364+1+r]-
((a[r+1])*s1[r+2]*s2[r+2]))}
}
if(onestepfnum>1819){
init6<-as.vector(data)[1:((initdatayear+5)*364)]
ts26<-ts(init6,start=1,frequency=7)
d26<-decompose(ts26,type="multiplicative",filter=NULL)
s26<-(as.vector(d26$seasonal))[1:364]
s2[1821:2184]<-s26
        for(j in 1819:(onestepfnum-1)){
a[j+1]<-lambda1*((as.vector(data)[initdatayear*364+1+j])/(s1[j+1]*s2[j+1]))+(1-lambda1)*(a[j]*b[j])
b[j+1]<-lambda2*(a[j+1]/a[j])+(1-lambda2)*b[j]
s1[364+j+1]<-lambda3*((as.vector(data)[initdatayear*364+1+j])/(a[j+1]*s2[j+1]))+(1-lambda3)*s1[j+1]
onestepf[j+1]<-(a[j+1]*b[j+1]^1)*s1[j+2]*s2[j+2]+lambda4*(as.vector(data)[initdatayear*364+1+j]-
((a[j+1])*s1[j+2]*s2[j+2]))}
}
if(onestepfnum>2183){
init7<-as.vector(data)[1:((initdatayear+6)*364)]
ts27<-ts(init7,start=1,frequency=7)
d27<-decompose(ts27,type="multiplicative",filter=NULL)
s27<-(as.vector(d27$seasonal))[1:364]
s2[2185:2548]<-s27
        for(y in 2183:(onestepfnum-1)){
a[y+1]<-lambda1*((as.vector(data)[initdatayear*364+1+y])/(s1[y+1]*s2[y+1]))+(1-lambda1)*(a[y]*b[y])
b[y+1]<-lambda2*(a[y+1]/a[y])+(1-lambda2)*b[y]
s1[364+y+1]<-lambda3*((as.vector(data)[initdatayear*364+1+y])/(a[y+1]*s2[y+1]))+(1-lambda3)*s1[y+1]
onestepf[y+1]<-(a[y+1]*b[y+1]^1)*s1[y+2]*s2[y+2]+lambda4*(as.vector(data)[initdatayear*364+1+y]-
((a[y+1])*s1[y+2]*s2[y+2]))}
}
if(onestepfnum>2547){
init8<-as.vector(data)[1:((initdatayear+7)*364)]
ts28<-ts(init8,start=1,frequency=7)
d28<-decompose(ts28,type="multiplicative",filter=NULL)
s28<-(as.vector(d28$seasonal))[1:364]
s2[2549:2912]<-s28
        for(o in 2547:(onestepfnum-1)){
a[o+1]<-lambda1*((as.vector(data)[initdatayear*364+1+o])/(s1[o+1]*s2[o+1]))+(1-lambda1)*(a[o]*b[o])
b[o+1]<-lambda2*(a[o+1]/a[o])+(1-lambda2)*b[o]
s1[364+o+1]<-lambda3*((as.vector(data)[initdatayear*364+1+o])/(a[o+1]*s2[o+1]))+(1-lambda3)*s1[o+1]
```

```
onestepf[o+1]<-(a[o+1]*b[o+1]^1)*s1[o+2]*s2[o+2]+lambda4*(as.vector(data)[initdatayear*364+1+o]-
((a[o+1])*s1[o+2]*s2[o+2]))}
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
onestepf[i+onestepfnum]<-(a[onestepfnum]*b[onestepfnum]^(i+1))*s1[i+onestepfnum+1-
364]*s2[i+onestepfnum+1-364]+lambda4^(i)*(as.vector(data)[initdatayear*364+onestepfnum+1]-
((a[onestepfnum-1]*b[onestepfnum-1])*s1[onestepfnum+1-364]*s2[onestepfnum+1-364]))}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

A.3.3. Holts Exponential Smoothing

```
holtsexposmooth<-function(lambda,data,n.ahead,initdatayear){
onestepfnum=length(data)-initdatayear*364-1
lambda1<-lambda[1]
lambda2<-lambda[2]
init<-as.vector(data)[1:(initdatayear*364)]
ts<-ts(init,start=1,frequency=364)
d<-decompose(ts,type="multiplicative",filter=NULL)
t<-d$trend
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*(as.vector(data)[(initdatayear*364+1)])+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
b[1]<-lambda2*(a[1]-(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*initbeta1
onestepf[1]<-a[1]+b[1]*1
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*(as.vector(data)[initdatayear*364+1+m])+(1-lambda1)*(a[m]+b[m])
b[m+1]<-lambda2*(a[m+1]-a[m])+(1-lambda2)*b[m]
onestepf[m+1]<-a[m+1]+b[m+1]*1
```

```
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
onestepf[i+onestepfnum]<-a[onestepfnum]+b[onestepfnum]*(i+1)
}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

A.3.4. Multiple Seasonal Exponential Smoothing Methods

Multiple Additive Seasonal Additive Trend Exponential Smoothing

```
multipleasat<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
onestepfnum=length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="additive",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="additive",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])-(s1[1]+s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
b[1]<-lambda2*(a[1]-(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*initbeta1
```

```
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])-(a[1]+s2[1]))+(1-lambda3)*s1[1]
s2[8]<-lambda4*((as.vector(data)[initdatayear*364+1])-(a[1]+s1[1]))+(1-lambda4)*s2[1]
onestepf[1]<-(a[1]+b[1]*1)+s1[2]+s2[2]
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])-(s1[m+1]+s2[m+1]))+(1-
lambda1)*(a[m]+b[m])
b[m+1]<-lambda2*(a[m+1]-a[m])+(1-lambda2)*b[m]
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])-(a[m+1]+s2[m+1]))+(1-
lambda3)*s1[m+1]
s2[7+m+1]<-lambda4*((as.vector(data)[initdatayear*364+1+m])-(a[m+1]+s1[m+1]))+(1-lambda4)*s2[m+1]
onestepf[m+1]<-(a[m+1]+b[m+1]*1)+(s1[m+2]+s2[m+2])}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
onestepf[i+onestepfnum]<-(a[onestepfnum]+b[onestepfnum]*(i+1))+s1[i+onestepfnum+1-
364]+s2[i+onestepfnum+1-364]}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Multiple Additive Seasonal Additive Trend Exponential Smoothing With Autocorrelation Adjustment

```
multipleasatauto<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
lambda5<-lambda[5]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="additive",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="additive",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
```

```
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])-(s1[1]+s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
b[1]<-lambda2*(a[1]-(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*initbeta1
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])-(a[1]+s2[1]))+(1-lambda3)*s1[1]
s2[8]<-lambda4*((as.vector(data)[initdatayear*364+1])-(a[1]+s1[1]))+(1-lambda4)*s2[1]
onestepf[1]<-(a[1]+b[1]*1)+s1[2]+s2[2]+lambda5^1*(as.vector(data)[initdatayear*364+1]-
((a[1])+s1[2]+s2[2]))
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])-(s1[m+1]+s2[m+1]))+(1-
lambda1)*(a[m]+b[m])
b[m+1]<-lambda2*(a[m+1]-a[m])+(1-lambda2)*b[m]
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])-(a[m+1]+s2[m+1]))+(1-
lambda3)*s1[m+1]
s2[7+m+1]<-lambda4*((as.vector(data)[initdatayear*364+1+m])-(a[m+1]+s1[m+1]))+(1-lambda4)*s2[m+1]
onestepf[m+1]<-
(a[m+1]+b[m+1]*1)+(s1[m+2]+s2[m+2])+lambda5*(as.vector(data)[initdatayear*364+1+m]-
((a[m+1])+s1[m+2]+s2[m+2]))}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
onestepf[i+onestepfnum]<-(a[onestepfnum]+b[onestepfnum]*(i+1))+s1[i+onestepfnum+1-
364]+s2[i+onestepfnum+1-364]+lambda5^(i)*(as.vector(data)[initdatayear*364+onestepfnum+1]-
((a[onestepfnum-1]+b[onestepfnum-1])+s1[onestepfnum+1-364]+s2[onestepfnum+1-364]))}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Multiple Additive Seasonal Multiplicative Damped Trend Exponential Smoothing

```
multipleasdmt<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
```

```
lambda5<-lambda[5]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="additive",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="additive",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
e<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])-(s1[1]+s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda5
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda5
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])-(a[1]+s2[1]))+(1-lambda3)*s1[1]
s2[8]<-lambda4*((as.vector(data)[initdatayear*364+1])-(a[1]+s1[1]))+(1-lambda4)*s2[1]
onestepf[1]<-(a[1]*b[1]^lambda5)+s1[2]+s2[2]
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])-(s1[m+1]+s2[m+1]))+(1-
lambda1)*(a[m]*b[m]^lambda5)
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*b[m]^lambda5
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])-(a[m+1]+s2[m+1]))+(1-
lambda3)*s1[m+1]
s2[7+m+1]<-lambda4*((as.vector(data)[initdatayear*364+1+m])-(a[m+1]+s1[m+1]))+(1-lambda4)*s2[m+1]
onestepf[m+1]<-(a[m+1]*b[m+1]^lambda5)+(s1[m+2]+s2[m+2])}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
        for(z in 1:(i+1)){
e[z]<-(lambda5^(z))}
```

```
se<-sum(e)
onestepf[i+onestepfnum]<-(a[onestepfnum]*b[onestepfnum]^(se))+s1[i+onestepfnum+1-
364]+s2[i+onestepfnum+1-364]}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Multiple Additive Seasonal Multiplicative Damped Trend Exponential Smoothing With Autocorrelation Adjustment

```
multipleasdmtauto<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
lambda5<-lambda[5]
lambda6<-lambda[6]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="additive",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="additive",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
e<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])-(s1[1]+s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda6
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda6
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])-(a[1]+s2[1]))+(1-lambda3)*s1[1]
```

```
s2[8]<-lambda4*((as.vector(data)[initdatayear*364+1])-(a[1]+s1[1]))+(1-lambda4)*s2[1]
onestepf[1]<-(a[1]*b[1]^lambda6)+s1[2]+s2[2]+lambda5^1*( as.vector(data)[initdatayear*364+1]-((a[1])+
s1[2]+s2[2]))
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])-(s1[m+1]+s2[m+1]))+(1-
lambda1)*(a[m]*b[m]^lambda6)
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*(b[m]^lambda6)
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])-(a[m+1]+s2[m+1]))+(1-
lambda3)*s1[m+1]
s2[7+m+1]<-lambda4*((as.vector(data)[initdatayear*364+1+m])-(a[m+1]+s1[m+1]))+(1-lambda4)*s2[m+1]
onestepf[m+1]<-
(a[m+1]*(b[m+1]^lambda6))+(s1[m+2]+s2[m+2])+lambda5*(as.vector(data)[initdatayear*364+1+m]-
((a[m+1])+s1[m+2]+s2[m+2]))}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
        for(z in 1:(i+1)){
e[z]<-(lambda6^(z))}
se<-sum(e)
onestepf[i+onestepfnum]<-(a[onestepfnum]*b[onestepfnum]^(se))+s1[i+onestepfnum+1-
364]+s2[i+onestepfnum+1-364]+lambda5^(i)*(as.vector(data)[onestepfnum+initdatayear*364+1]-
((a[onestepfnum-1]*b[onestepfnum-1]^lambda6)+s1[onestepfnum+1-364]+s2[onestepfnum+1-364]))}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Multiple Additive Seasonal Multiplicative Trend Exponential Smoothing With Autocorrelation Adjustment

```
multipleasmtauto<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
lambda5<-lambda[5]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="additive",filter=NULL)
```

```
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="additive",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])-(s1[1]+s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])-(a[1]+s2[1]))+(1-lambda3)*s1[1]
s2[8]<-lambda4*((as.vector(data)[initdatayear*364+1])-(a[1]+s1[1]))+(1-lambda4)*s2[1]
onestepf[1]<-(a[1]*b[1]^1)+s1[2]+s2[2]+lambda5^1*( as.vector(data)[initdatayear*364+1]-
((a[1])+s1[2]+s2[2]))
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])-(s1[m+1]+s2[m+1]))+(1-
lambda1)*(a[m]*b[m])
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*b[m]
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])-(a[m+1]+s2[m+1]))+(1-
lambda3)*s1[m+1]
s2[7+m+1]<-lambda4*((as.vector(data)[initdatayear*364+1+m])-(a[m+1]+s1[m+1]))+(1-lambda4)*s2[m+1]
onestepf[m+1]<-
(a[m+1]*b[m+1]^1)+(s1[m+2]+s2[m+2])+lambda5*(as.vector(data)[initdatayear*364+1+m]-
((a[m+1])+s1[m+2]+s2[m+2]))}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
onestepf[i+onestepfnum]<-(a[onestepfnum]*b[onestepfnum]^(i+1))+s1[i+onestepfnum+1-
364]+s2[i+onestepfnum+1-364]+lambda5^(i)*(as.vector(data)[onestepfnum+initdatayear*364+1]-
((a[onestepfnum-1]*b[onestepfnum-1])+s1[onestepfnum+1-364]+s2[onestepfnum+1-364]))}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Multiple Multiplicative Seasonal Additive Trend Exponential Smoothing

```
multiplemsat<-function(lambda,data,n.ahead,initdatayear){
onestepfnum=length(data)-initdatayear*364-1
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="multiplicative",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="multiplicative",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[(initdatayear*364+1)])/(s1[1]*s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
b[1]<-lambda2*(a[1]-(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*initbeta1
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])/(a[1]*s2[1]))+(1-lambda3)*s1[1]
s2[8]<-lambda4*((as.vector(data)[(initdatayear*364+1)])/(a[1]*s1[1]))+(1-lambda4)*s2[1]
onestepf[1]<-(a[1]+b[1]*1)*s1[2]*s2[2]
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])/(s1[m+1]*s2[m+1]))+(1-
lambda1)*(a[m]+b[m])
b[m+1]<-lambda2*(a[m+1]-a[m])+(1-lambda2)*b[m]
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s2[m+1]))+(1-
lambda3)*s1[m+1]
s2[7+m+1]<-lambda4*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s1[m+1]))+(1-lambda4)*s2[m+1]
onestepf[m+1]<-(a[m+1]+b[m+1]*1)*s1[m+2]*s2[m+2]}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
```

```
        for(i in 1:n.ahead){
onestepf[i+onestepfnum]<-(a[onestepfnum]+b[onestepfnum]*(i+1))*s1[i+onestepfnum+1-
364]*s2[i+onestepfnum+1-364]}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Multiple Multiplicative Seasonal Additive Trend Exponential Smoothing With Autocorrelation Adjustment

```
multiplemsatauto<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
lambda5<-lambda[5]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="multiplicative",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="multiplicative",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])/(s1[1]*s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
b[1]<-lambda2*(a[1]-(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*initbeta1
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])/(a[1]*s2[1]))+(1-lambda3)*s1[1]
s2[8]<-lambda4*((as.vector(data)[initdatayear*364+1])/(a[1]*s1[1]))+(1-lambda4)*s2[1]
onestepf[1]<-(a[1]+b[1]*1)*s1[2]*s2[2]+lambda5^1*( as.vector(data)[initdatayear*364+1]-((a[1])*
s1[2]*s2[2]))
        for(m in 1:(onestepfnum-1)){
```

```
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])/(s1[m+1]*s2[m+1]))+(1-
lambda1)*(a[m]+b[m])
b[m+1]<-lambda2*(a[m+1]-a[m])+(1-lambda2)*b[m]
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s2[m+1]))+(1-
lambda3)*s1[m+1]
s2[7+m+1]<-lambda4*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s1[m+1]))+(1-lambda4)*s2[m+1]
onestepf[m+1]<-(a[m+1]+b[m+1]*1)*s1[m+2]*s2[m+2]+lambda5*(as.vector(data)[initdatayear*364+1+m]-
((a[m+1])*s1[m+2]*s2[m+2]))}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
onestepf[i+onestepfnum]<-(a[onestepfnum]+b[onestepfnum]*(i+1))*s1[i+onestepfnum+1-
364]*s2[i+onestepfnum+1-364]+lambda5^(i)*(as.vector(data)[initdatayear*364+onestepfnum+1]-
((a[onestepfnum-1]+b[onestepfnum-1])*s1[onestepfnum+1-364]*s2[onestepfnum+1-364]))}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Multiple Multiplicative Seasonal Multiplicative Damped Trend Exponential Smoothing

```
multiplemsdmt<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
lambda5<-lambda[5]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="multiplicative",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="multiplicative",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
```

```
onestepf<-vector(mode="logical")
e<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])/(s1[1]*s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda5
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda5
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])/(a[1]*s2[1]))+(1-lambda3)*s1[1]
s2[8]<-lambda4*((as.vector(data)[initdatayear*364+1])/(a[1]*s1[1]))+(1-lambda4)*s2[1]
onestepf[1]<-(a[1]*b[1]^lambda5)*s1[2]*s2[2]
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])/(s1[m+1]*s2[m+1]))+(1-
lambda1)*(a[m]*b[m]^lambda5)
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*b[m]^lambda5
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s2[m+1]))+(1-
lambda3)*s1[m+1]
s2[7+m+1]<-lambda4*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s1[m+1]))+(1-lambda4)*s2[m+1]
onestepf[m+1]<-(a[m+1]*b[m+1]^lambda5)*s1[m+2]*s2[m+2]}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
        for(z in 1:(i+1)){
e[z]<-(lambda5^(z))}
se<-sum(e)
onestepf[i+onestepfnum]<-(a[onestepfnum]*b[onestepfnum]^(se))*s1[i+onestepfnum+1-
364]*s2[i+onestepfnum+1-364]}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Multiple Multiplicative Seasonal Damped Multiplicative Trend Exponential Smoothing With Autocorrelation Adjustment

```
multiplemsdmtauto<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
```

```
lambda4<-lambda[4]
lambda5<-lambda[5]
lambda6<-lambda[6]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="multiplicative",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="multiplicative",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
e<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])/(s1[1]*s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda6
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))^lambda6
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])/(a[1]*s2[1]))+(1-lambda3)*s1[1]
s2[8]<-lambda4*((as.vector(data)[initdatayear*364+1])/(a[1]*s1[1]))+(1-lambda4)*s2[1]
onestepf[1]<-(a[1]*b[1]^lambda6)*s1[2]*s2[2]+lambda5^1*( as.vector(data)[initdatayear*364+1]-((a[1])*
s1[2]*s2[2]))
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])/(s1[m+1]*s2[m+1]))+(1-
lambda1)*(a[m]*b[m]^lambda6)
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*b[m]^lambda6
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s2[m+1]))+(1-
lambda3)*s1[m+1]
s2[7+m+1]<-lambda4*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s1[m+1]))+(1-lambda4)*s2[m+1]
onestepf[m+1]<-
(a[m+1]*b[m+1]^lambda6)*s1[m+2]*s2[m+2]+lambda5*(as.vector(data)[initdatayear*364+1+m]-
((a[m+1])*s1[m+2]*s2[m+2]))}
if(n.ahead==0){
```

```
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
        for(z in 1:(i+1)){
e[z]<-(lambda6^(z))}
se<-sum(e)
onestepf[i+onestepfnum]<-(a[onestepfnum]*b[onestepfnum]^(se))*s1[i+onestepfnum+1-
364]*s2[i+onestepfnum+1-364]+lambda5^(i)*(as.vector(data)[onestepfnum+initdatayear*364+1]-
((a[onestepfnum-1]*b[onestepfnum-1]^lambda6)*s1[onestepfnum+1-364]*s2[onestepfnum+1-364]))}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

Multiplicative Multiple Seasonal Multiplicative Trend Exponential Smoothing With Autocorrelation Adjustment

```
multiplemsmtauto<-function(lambda,data,n.ahead,initdatayear){
lambda1<-lambda[1]
lambda2<-lambda[2]
lambda3<-lambda[3]
lambda4<-lambda[4]
lambda5<-lambda[5]
onestepfnum<-length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
ts1<-ts(init,start=1,frequency=364)
d1<-decompose(ts1,type="multiplicative",filter=NULL)
ts2<-ts(init,start=1,frequency=7)
d2<-decompose(ts2,type="multiplicative",filter=NULL)
t<-d1$trend
s1<-(as.vector(d1$seasonal))[1:364]
s2<-(as.vector(d2$seasonal))[1:364]
l<-c(1:(initdatayear*364))
g<-lm(as.vector(t)~l)
a<-vector(mode="logical")
b<-vector(mode="logical")
onestepf<-vector(mode="logical")
initbeta0<-as.vector(g$coefficients)[1]
initbeta1<-as.vector(g$coefficients)[2]
a[1]<-lambda1*((as.vector(data)[initdatayear*364+1])/(s1[1]*s2[1]))+(1-
lambda1)*(initbeta0+initbeta1*initdatayear*364+initbeta1)
```

```
b[1]<-lambda2*(a[1]/(initbeta0+initbeta1*initdatayear*364))+(1-lambda2)*
((initbeta0+initbeta1*initdatayear*364+initbeta1)/(initbeta0+initbeta1*initdatayear*364))
s1[365]<-lambda3*((as.vector(data)[initdatayear*364+1])/(a[1]*s2[1]))+(1-lambda3)*s1[1]
s2[8]<-lambda4*((as.vector(data)[initdatayear*364+1])/(a[1]*s1[1]))+(1-lambda4)*s2[1]
onestepf[1]<-(a[1]*b[1]^1)*s1[2]*s2[2]+lambda5^1*( as.vector(data)[initdatayear*364+1]-((a[1])*
s1[2]*s2[2]))
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda1*((as.vector(data)[initdatayear*364+1+m])/(s1[m+1]*s2[m+1]))+(1-
lambda1)*(a[m]*b[m])
b[m+1]<-lambda2*(a[m+1]/a[m])+(1-lambda2)*b[m]
s1[364+m+1]<-lambda3*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s2[m+1]))+(1-
lambda3)*s1[m+1]
s2[7+m+1]<-lambda4*((as.vector(data)[initdatayear*364+1+m])/(a[m+1]*s1[m+1]))+(1-lambda4)*s2[m+1]
onestepf[m+1]<-(a[m+1]*b[m+1]^1)*s1[m+2]*s2[m+2]+lambda5*(as.vector(data)[initdatayear*364+1+m]-
((a[m+1])*s1[m+2]*s2[m+2]))}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
onestepf[i+onestepfnum]<-(a[onestepfnum]*b[onestepfnum]^(i+1))*s1[i+onestepfnum+1-
364]*s2[i+onestepfnum+1-364]+lambda5^(i)*(as.vector(data)[onestepfnum+initdatayear*364+1]-
((a[onestepfnum-1]*b[onestepfnum-1])*s1[onestepfnum+1-364]*s2[onestepfnum+1-364]))}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

A.3.5. Simple Exponential Smoothing

```
simpleexposmooth<-function(lambda,data,n.ahead,initdatayear){
onestepfnum=length(data)-initdatayear*364-1
init<-as.vector(data)[1:(initdatayear*364)]
initbeta0<-mean(init)
a<-vector(mode= "logical")
onestepf<-vector(mode= "logical")
a[1]<-lambda*(as.vector(data)[initdatayear*364+1])+(1-lambda)*initbeta0
onestepf[1]<-a[1]
        for(m in 1:(onestepfnum-1)){
a[m+1]<-lambda*(as.vector(data)[initdatayear*364+1+m])+(1-lambda)*a[m]
onestepf[m+1]<-a[m+1]
```

```
}
if(n.ahead==0){
return(mean((onestepf-as.vector(data)[(initdatayear*364+2):(initdatayear*364+1+onestepfnum)])^2))}
else{
        for(i in 1:n.ahead){
 onestepf[i+onestepfnum]<-a[onestepfnum]}
return(onestepf[(onestepfnum+1):(onestepfnum+n.ahead)])
}
}
```

A.4. R Codes For Sarima_Weekday Method

```
resm1_pass<-matrix(rep(0,(simdays_pass*(length(data_pass)-simdays_pass-n.ahead-
2))),ncol=(length(data_pass)-simdays_pass-n.ahead-2))
a1_pass<-matrix(rep(0,(simdays_pass*(length(data_pass)-simdays_pass-n.ahead-2))),ncol=simdays_pass)
resm_pass<-matrix(rep(0,(simdays_pass*n.ahead)),ncol=n.ahead)
a_pass<-matrix(rep(0,(simdays_pass*n.ahead)),ncol=simdays_pass)
        for(i in 1:simdays_pass){
wds<-matrix((data_pass[(i+3):(length(data_pass)-simdays_pass-n.ahead+i)]),7,byrow=FALSE)
#wds<-week day separation
sunday<-wds[1,]
monday<-wds[2,]
tuesday<-wds[3,]
wednesday<-wds[4,]
thursday<-wds[5,]
friday<-wds[6,]
saturday<-wds[7,]
sun<-ts(sunday,start=1,frequency=52)
b<-
arima(sun,order=c(2,0,0),seasonal=list(order=c(1,1,0)),xreg=NULL,method="ML",optim.method="BFGS",o
ptim.control=list(maxit=10000),kappa=1e6)
sundaypredict<-predict(b,n.ahead=(n.ahead/7))
mon<-ts(monday,start=1,frequency=52)
b2<-
arima(mon,order=c(2,0,0),seasonal=list(order=c(1,1,0)),xreg=NULL,method="ML",optim.method="BFGS",
optim.control=list(maxit=10 000),kappa=1e6)
mondaypredict<-predict(b2,n.ahead=(n.ahead/7))
tue<-ts(tuesday,start=1,frequency=52)
```

```
b3<-
arima(tue,order=c(2,0,0),seasonal=list(order=c(1,1,0)),xreg=NULL,method="ML",optim.method="BFGS",o
ptim.control=list(maxit=10 000),kappa=1e6)
tuesdaypredict<-predict(b3,n.ahead=(n.ahead/7))
wed<-ts(wednesday,start=1,frequency=52)
b4<-
arima(wed,order=c(2,0,0),seasonal=list(order=c(1,1,0)),xreg=NULL,method="ML",optim.method="BFGS",
optim.control=list(maxit=10 000),kappa=1e6)
wednesdaypredict<-predict(b4,n.ahead=(n.ahead/7))
thurs<-ts(thursday,start=1,frequency=52)
b5<-
arima(thurs,order=c(2,0,0),seasonal=list(order=c(1,1,0)),xreg=NULL,method="ML",optim.method="BFGS",
optim.control=list(maxit=100 0),kappa=1e6)
thursdaypredict<-predict(b5,n.ahead=(n.ahead/7))
fri<-ts(friday,start=1,frequency=52)
b6<-
arima(fri,order=c(2,0,0),seasonal=list(order=c(1,1,0)),xreg=NULL,method="ML",optim.method="BFGS",op
tim.control=list(maxit=10 000),kappa=1e6)
fridaypredict<-predict(b6,n.ahead=(n.ahead/7))
sat<-ts(saturday,start=1,frequency=52)
b7<-
arima(sat,order=c(2,0,0),seasonal=list(order=c(1,1,0)),xreg=NULL,method="ML",optim.method="BFGS",o
ptim.control=list(maxit=10 000),kappa=1e6)
saturdaypredict<-predict(b7,n.ahead=(n.ahead/7))
a1_pass[,i]<-
as.vector(matrix(c(as.vector(fitted(b)),as.vector(fitted(b2)),as.vector(fitted(b3)),as.vector(fitted(b4)),as.vector
(fitted(b5)),as.vector(fitted(b6)),as.vector(fitted(b7))),7,byrow=TRUE))
a_pass[,i]<-
as.vector(matrix(c(as.vector(sundaypredict$pred),as.vector(mondaypredict$pred),as.vector(tuesdaypredict$pr
ed),as.vector(wednesdaypredict$pred),as.vector(thursdaypredict$pred),as.vector(fridaypredict$pred),as.vecto
r(saturdaypredict$pred)),7,byrow=TRUE))
resm_pass[i,]<-a_pass[,i]
resm1_pass[i,]<-a1_pass[,i]
}
```

A.5. General R Codes

```
estimate_general<-function(startingvalues,dat,MSEfunc,initdata,...){
```

```
#this function finds the optimum parameters giving the minimum MSE of one step ahead forecasts of
onestepfnum times. It starts finding optimum parameters beginning from the given starting values parameter
vector.
#starting values… are the startin value of parameters used in estimate_singlemsat function
#dat… is the data used in optim and singlemsmt function
#n.ahead… provides to determine the usage of MSE results in singlemsat function.
optim<-optim(startingvalues,MSEfunc,data=dat,...,n.ahead=0,initdatayear=initdata,method="L-BFGS-
B",lower=1e-9,upper=1)
return(optim$par)
}

estimate_generalnaive<-function(startingvalues,dat,new,MSEfunc,...){
#this function finds the optimum parameters giving the minimum MSE of one step ahead forecasts of
onestepfnum times. It starts finding optimum parameters beginning from the given starting values parameter
vector.
#starting values… are the starting value of parameters used in estimate_general function
#dat… is the data used in optim and singlemsat function
#n.ahead… provides to determine the usage of MSE results in singlemsat function.
optim<-optim(startingvalues,MSEfunc,dv=dat,new1=new,...,n.ahead=200,method="L-BFGS-B",lower=1e-
6,upper=1)
return(optim$par)
}

historicsimulationnaive<-function(data,n.ahead,onestepfnumm,simdays,parnum,MSEfunc,...){
mat<-matrix(rep(0,(n.ahead*simdays)),ncol=n.ahead)
optmat<-matrix(rep(0,(parnum*simdays)),ncol=parnum)
length<-length(data)
        for(n in 1:simdays){
MSEdata<-data[(length-simdays-onestepfnumm-n.ahead+n+1):(length-simdays-n.ahead+n)]
newdata1<-data[1:(length-simdays-onestepfnumm-n.ahead+n)]
newdata2<-data[1:(length-simdays-n.ahead+n)]
if(n==1){
optmat[n,]<-estimate_general(c(1e-6,1e-6,1e-6),newdata1,MSEdata,MSEfunc=MSEfunc,...)}
else{
optmat[n,]<-estimate_general(c(optmat[(n-1),1],optmat[(n-1),2],optmat[(n-
1),3]),newdata1,MSEdata,MSEfunc=MSEfunc,...)}
mat[n,]<-MSEfunc(c(optmat[n,1],optmat[n,2],optmat[n,3]),dv=newdata2,MSEdata,n.ahead=n.ahead,...)
}
return(mat)
}
```

```
historicsimulationauto<-function(data,n.ahead,initdatayear,simdays,parnum,func, ...){
#this function prepares the part of the data that is going to be used according to simulation
number,n.ahead,initdatayear and returns predicted values matrix.
#n.ahead…specifies how many steps ahead to predict
#initdatayear…start periods used in autodetection of start values, it should be at least 2 periods
#simdays…number of simulation days
#parnum…parameter number of the automaticmsat function
mat<-matrix(rep(0,(n.ahead*simdays)),ncol=n.ahead)
optmat<-matrix(rep(0,(parnum*simdays)),ncol=parnum)
length<-length(data)
cutoffvalue<-(length-n.ahead-simdays-(initdatayear*364)+1)
if(cutoffvalue>0){
newdata<-as.vector(data)[-(1:cutoffvalue)]}
if (cutoffvalue==0){newdata<-data}
if(cutoffvalue<0){print("change parameters or decrease simdays")}
        for(n in 1:simdays){
lastnewdata<-newdata[n:(initdatayear*364-1+n)]
if(n==1){
optmat[n,]<-func(c(1e-9,1e-9,1e-9),lastnewdata,0,initdatayear)}
else{
optmat[n,]<-func(c(optmat[(n-1),1],optmat[(n-1),2],optmat[(n-1),3]),lastnewdata,0,initdatayear)}
mat[n,]<-func(c(optmat[n,1],optmat[n,2],optmat[n,3]),lastnewdata,n.ahead,initdatayear)
}
return(mat)
}

historicsimulation1<-function(data,n.ahead,onestepfnumm,initdatayear,simdays,parnum,MSEfunc,...){
#this function prepares the part of the data that is going to be used according to simulation number, onestep
ahead forecast number, initialization period and forecast numbers and returns a matrix of historic forecasts by
optimizing parameters every 20th day.
#n.ahead…specifies how many steps ahead to predict
#onestepfnum…number of one step ahead forecasts
#initdatayear…start periods used in autodetection of start values, it should be at least 2 periods
#simdays…number of simulation days
#parnum…parameter number of the singlemsmt function
mat<-matrix(rep(0,(n.ahead*simdays)),ncol=n.ahead)
optmat<-matrix(rep(0,(parnum*simdays)),ncol=parnum)
length<-length(data)
cutoffvalue<-(length-n.ahead-onestepfnumm-simdays-(initdatayear*364))
if(cutoffvalue>0){
```

```
newdata<-as.vector(data)[-(1:cutoffvalue)]}
if (cutoffvalue==0){newdata<-data}
if(cutoffvalue<0){print("change parameters or decrease simdays")}
        for(n in 1:simdays){
lastnewdata<-newdata[n:(onestepfnumm+(initdatayear*364)+n)]
if(n==1){
optmat[n,]<-estimate_general(c(1e-9),lastnewdata,MSEfunc=MSEfunc,initdatayear,...)}
else{if(n%%20==1){
optmat[n,]<-estimate_general(c(optmat[(n-1),1]),lastnewdata,MSEfunc=MSEfunc,initdatayear,...)}}
mat[n,]<-MSEfunc(c(optmat[n,1]),data=lastnewdata,n.ahead,initdatayear,...)
}
return(mat)
}

historicsimulation2<-function(data,n.ahead,onestepfnumm,initdatayear,simdays,parnum,MSEfunc,...){
#optimization is done every day.
mat<-matrix(rep(0,(n.ahead*simdays)),ncol=n.ahead)
optmat<-matrix(rep(0,(parnum*simdays)),ncol=parnum)
length<-length(data)
cutoffvalue<-(length-n.ahead-onestepfnumm-simdays-(initdatayear*364))
if(cutoffvalue>0){
newdata<-as.vector(data)[-(1:cutoffvalue)]}
if (cutoffvalue==0){newdata<-data}
if(cutoffvalue<0){print("change parameters or decrease simdays")}
        for(n in 1:simdays){
lastnewdata<-newdata[n:(onestepfnumm+(initdatayear*364)+n)]
if(n==1){
optmat[n,]<-estimate_general(c(1e-9,1e-9),lastnewdata,MSEfunc=MSEfunc,initdatayear,...)}
else{
optmat[n,]<-estimate_general(c(optmat[(n-1),1],optmat[(n-
1),2]),lastnewdata,MSEfunc=MSEfunc,initdatayear,...)}
mat[n,]<-MSEfunc(c(optmat[n,1],optmat[n,2]),data=lastnewdata,n.ahead,initdatayear,...)
}
return(mat)
}

historicsimulation3<-function(data,n.ahead,onestepfnumm,initdatayear,simdays,parnum,MSEfunc,...){
mat<-matrix(rep(0,(n.ahead*simdays)),ncol=n.ahead)
optmat<-matrix(rep(0,(parnum*simdays)),ncol=parnum)
length<-length(data)
```

```
cutoffvalue<-(length-n.ahead-onestepfnumm-simdays-(initdatayear*364))
if(cutoffvalue>0){
newdata<-as.vector(data)[-(1:cutoffvalue)]}
if (cutoffvalue==0){newdata<-data}
if(cutoffvalue<0){print("change parameters or decrease simdays")}
        for(n in 1:simdays){
lastnewdata<-newdata[n:(onestepfnumm+(initdatayear*364)+n)]
if(n==1){
optmat[n,]<-estimate_general(c(1e-9,1e-9,1e-9),lastnewdata,MSEfunc=MSEfunc,initdatayear,...)}
else{
optmat[n,]<-estimate_general(c(optmat[(n-1),1],optmat[(n-1),2],optmat[(n-
1),3]),lastnewdata,MSEfunc=MSEfunc,initdatayear,...)}
mat[n,]<-MSEfunc(c(optmat[n,1],optmat[n,2],optmat[n,3]),data=lastnewdata,n.ahead,initdatayear,...)
}
return(mat)
}

historicsimulation4<-function(data,n.ahead,onestepfnumm,initdatayear,simdays,parnum,MSEfunc,...){
mat<-matrix(rep(0,(n.ahead*simdays)),ncol=n.ahead)
optmat<-matrix(rep(0,(parnum*simdays)),ncol=parnum)
length<-length(data)
cutoffvalue<-(length-n.ahead-onestepfnumm-simdays-(initdatayear*364))
if(cutoffvalue>0){
newdata<-as.vector(data)[-(1:cutoffvalue)]}
if (cutoffvalue==0){newdata<-data}
if(cutoffvalue<0){print("change parameters or decrease simdays")}
        for(n in 1:simdays){
lastnewdata<-newdata[n:(onestepfnumm+(initdatayear*364)+n)]
if(n==1){
optmat[n,]<-estimate_general(c(1e-9,1e-9,1e-9,1e-9),lastnewdata,MSEfunc=MSEfunc,initdatayear,...)}
else{
optmat[n,]<-estimate_general(c(optmat[(n-1),1],optmat[(n-1),2],optmat[(n-1),3],optmat[(n-
1),4]),lastnewdata,MSEfunc=MSEfunc,initdatayear,...)}
mat[n,]<-
MSEfunc(c(optmat[n,1],optmat[n,2],optmat[n,3],optmat[n,4]),data=lastnewdata,n.ahead,initdatayear,...)
}
return(mat)
}

historicsimulation5<-function(data,n.ahead,onestepfnumm,initdatayear,simdays,parnum,MSEfunc,...){
```

```
mat<-matrix(rep(0,(n.ahead*simdays)),ncol=n.ahead)
optmat<-matrix(rep(0,(parnum*simdays)),ncol=parnum)
length<-length(data)
cutoffvalue<-(length-n.ahead-onestepfnumm-simdays-(initdatayear*364))
if(cutoffvalue>0){
newdata<-as.vector(data)[-(1:cutoffvalue)]}
if (cutoffvalue==0){newdata<-data}
if(cutoffvalue<0){print("change parameters or decrease simdays")}
        for(n in 1:simdays){
lastnewdata<-newdata[n:(onestepfnumm+(initdatayear*364)+n)]
if(n==1){
optmat[n,]<-estimate_general(c(1e-9,1e-9,1e-9,1e-9,1e-9),lastnewdata,MSEfunc=MSEfunc,initdatayear,...)}
else{
optmat[n,]<-estimate_general(c(optmat[(n-1),1],optmat[(n-1),2],optmat[(n-1),3],optmat[(n-1),4],optmat[(n-
1),5]),lastnewdata,MSEfunc=MSEfunc,initdatayear,...)}
mat[n,]<-
MSEfunc(c(optmat[n,1],optmat[n,2],optmat[n,3],optmat[n,4],optmat[n,5]),data=lastnewdata,n.ahead,initdata
year,...)
}
return(mat)
}

historicsimulation6<-function(data,n.ahead,onestepfnumm,initdatayear,simdays,parnum,MSEfunc,...){
mat<-matrix(rep(0,(n.ahead*simdays)),ncol=n.ahead)
optmat<-matrix(rep(0,(parnum*simdays)),ncol=parnum)
length<-length(data)
cutoffvalue<-(length-n.ahead-onestepfnumm-simdays-(initdatayear*364))
if(cutoffvalue>0){
newdata<-as.vector(data)[-(1:cutoffvalue)]}
if (cutoffvalue==0){newdata<-data}
if(cutoffvalue<0){print("change parameters or decrease simdays")}
        for(n in 1:simdays){
lastnewdata<-newdata[n:(onestepfnumm+(initdatayear*364)+n)]
if(n==1){
optmat[n,]<-estimate_general(c(1e-9,1e-9,1e-9,1e-9,1e-9,1e-
9),lastnewdata,MSEfunc=MSEfunc,initdatayear,...)}
else{
optmat[n,]<-estimate_general(c(optmat[(n-1),1],optmat[(n-1),2],optmat[(n-1),3],optmat[(n-1),4],optmat[(n-
1),5],optmat[(n-1),6]),lastnewdata,MSEfunc=MSEfunc,initdatayear,...)}
```

```
mat[n,]<-
MSEfunc(c(optmat[n,1],optmat[n,2],optmat[n,3],optmat[n,4],optmat[n,5],optmat[n,6]),data=lastnewdata,n.ah
ead,initdatayear,...)
}
return(mat)
}

naivesimulationfunction<-function(data,simdays,n.ahead){
l<-length(data)
resm<-matrix(rep(0,(simdays*n.ahead)),ncol=n.ahead)
realm<-matrix(rep(0,(simdays*n.ahead)),ncol=n.ahead)
        for(a in 1:simdays){
        for(b in 1:n.ahead){
realm[a,b]<-as.vector(data)[l-simdays-n.ahead+a+b]
}
}
        for(k in 1:simdays){
resm[k,]<-naive(data[1:(l-simdays-n.ahead+k)],n.ahead)
}
return(resm)
}

MAPE_function<-function(data,n.ahead,simdays,forecastm,...){
MAPE<-vector(mode= "logical")
realm<-matrix(rep(0,(simdays*n.ahead)),ncol=n.ahead)
        for(a in 1:simdays){
        for(b in 1:n.ahead){
realm[a,b]<-as.vector(data)[length(data)-n.ahead-simdays+a+b]
}}
APE<-abs((forecastm-realm)/realm)
        for(j in 1:n.ahead){
MAPE[j]<-mean(APE[,j])}
return(MAPE)
}

MAD_function<-function(data,n.ahead,simdays,forecastm,...){
MAD<-vector(mode= "logical")
realm<-matrix(rep(0,(simdays*n.ahead)),ncol=n.ahead)
        for(a in 1:simdays){
        for(b in 1:n.ahead){
```

```
realm[a,b]<-as.vector(data)[length(data)-n.ahead-simdays+a+b]
}}
AD<-abs(forecastm-realm)
        for(j in 1:n.ahead){
MAD[j]<-mean(AD[,j])}
return(MAD)
}

MSE_function<-function(data,n.ahead,simdays,forecastm,...){
MSE<-vector(mode= "logical")
realm<-matrix(rep(0,(simdays*n.ahead)),ncol=n.ahead)
        for(a in 1:simdays){
        for(b in 1:n.ahead){
realm[a,b]<-as.vector(data)[length(data)-n.ahead-simdays+a+b]
}}
SE<-(forecastm-realm)^2
        for(j in 1:n.ahead){
MSE[j]<-sum(SE[,j])/(simdays)}
return(MSE)
}

AE_function<-function(data,n.ahead,simdays,forecastm,...){
AE<-vector(mode= "logical")
realm<-matrix(rep(0,(simdays*n.ahead)),ncol=n.ahead)
        for(a in 1:simdays){
        for(b in 1:n.ahead){
realm[a,b]<-as.vector(data)[length(data)-n.ahead-simdays+a+b]
}}
AE<-abs(forecastm-realm)
return(AE)
}
```

REFERENCES

Bowerman, B. L., O'Connell, R. T., Koehler, A. B., 2005, *Forecasting, Time Series, and Regression An Applied Approach* (4th ed.) Belmont: Thomson Books/Cole.

Cancelo, J. R., Espasa, A., Grafe, R., 2008, "Forecasting the electricity load from one day to one week ahead for the Spanish system operator", *International Journal of Forecasting, 24,* 588-602.

Conejo, A. J., Contreras, J.,Espinola, R.,Plazas, M. A., 2005, "Forecasting electricity prices for a day ahead pool-based electric energy market", *International Journal of Forecasting, 21,* 435-462.

Cranage, D., 2003, "Practical time series forecasting for the hospitality manager", *International Journal of Contemporary Hospitality Management,15,2,* pp. 86-93.

Gould, P. G., Koehler, A. B., Ord, J. K., Synder, R. D., Hyndman R. J., Araghi, F. V., 2008, "Forecasting time series with multiple seasonal patterns", *European Journal of Operational Research, 191,* 207-222, Elsevier.

Grubb, H., Mason, A., 2001, "Long lead-time forecasting of UK air passengers by Holt-Winters methods with damped trend", *International Journal of Forecasting, 17,* 71-82, Elsevier.

Hanke, J. E., Wichern, D. W., 2009, *Business Forecasting* (9th ed.), New Jersey: Pearson Prentice Hall.

Hippert, H. S., Bunn, D. W., & Souza, R. C. (2005). "Large neural networks for electricity load forecasting: are they overfitted?", *International Journal of Forecasting, 21*, 425-434.

Hyndman, R. J., Khandakar, Y., 2008, "Automatic Time Series Forecasting: The forecast package for R", *Journal of Statistical Software, 27,3,* 1-19.

Hyndman, R. J., Koehler, A. B., Ord, J. K. & Synder, R. D., 2008, *Forecasting with Exponential Smoothing, The State Space Approach,* Berlin: Springer.

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

Kulahci, M., Bisgaard, S., 2008, "Quality Quandaries: Box-Cox Transformations and Time Series Modeling", *Quality Engineering,20,4,* pp. 376-388,516-523.

Livera, A. M., Hyndman, R. J., 2009, *Forecasting Time Series with Complex Seasonal Patterns using Exponential Smoothing*, http://www.buseco.monash.edu.au/depts/ebs/pubs /wpapers

Montgomery, D. C.,Jennings, C. L.,Kulahci, M., 2008, *Introduction to Time Series Analysis and Forecasting,* New Jersey: John Wiley & Sons, Inc.

Newbold, P., Bos, T., 1994, *Introductory Business and Economics Forecasting*(2nd ed.),Ohio: International Thomson Publishing.

Saunders, J. A., Sharp, J. A., Witt, S. F., 1987, *Practical Business Forecasting,* Brookfield: Gower.

Taylor, J. W., Menezes, L. M. &McSharry, P. E., 2006, "A comparison of univariate methods for forecasting electricity demand up to a day ahead", *International Journal of Forecasting,22,*1-16, Elsevier.

Taylor, J., 2003, "Exponential smoothing with a damped multiplicative trend", *International Journal of Forecasting, 19,* 715-725, Elsevier.

Taylor, J.W., 2010, "Triple Seasonal Methods for short term electricity demand forecasting", *European Journal of Operational Research, 204,* 139-152.

Tong, L. I., Liang, Y. H., 2002, "Forecasting field failure data for repairable systems using neural networks and sarima model", *International journal of Quality and Reliability Management,22,4,* 410-420.

Weatherford, L. R., Kimes, S.E., 2003, "A comparison of methods for hotel revenue management", *International Journal of Forecasting, 19,* 401-415.

Wilson, J. H., Keating, B., 2002, *Business Forecasting with Accompanying Excel-Based ForecastX Software* (4th ed.), New York: McGraw-Hill.