INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa · University of Ottawa

PERMISSION DE REPRODUIRE ET DE DISTRIBUER LA THÈSE

PERMISSION TO REPRODUCE AND DISTRIBUTE THE THESIS

NOM DE L'AUTEUR / NAME OF AUTHOR:	WANG, Sudan
ADRESSE POSTALE / MAILING ADDRESS:	609-101 BOULEVARD SACRÉ COEUR HULL QC J8X1C7
GRADE / DEGREE: M.Sc. (Systems Science)	**ANNÉE D'OBTENTION / YEAR GRANTED** 2003
TITRE DE LA THÈSE / TITLE OF THESIS:	ISSUES IN ADDING SPECIAL PURPOSE FUNCTIONS TO FREE SCIENTIFIC OR STATISTICAL SOFTWARE

L'auteur permet, par la présente, la consultation et le prêt de cette thèse en conformité avec les règlements établis par le bibliothécaire en chef de l'Université d'Ottawa. L'auteur autorise aussi l'Université d'Ottawa, ses successeurs et cessionnaires, à reproduire cet exemplaire par photograhie ou photocopie pour fins de prêt ou de vente au prix coûtant aux bibliothèques ou aux chercheurs qui en feront la demande.

Les droits de publication par tout autre moyen et pour vente au public demeureront la propriété de l'auteur de la thèse sous réserve des règlements de l'Université d'Ottawa en matière de publication de thèses.

The author hereby permits the consultation and the lending of this thesis pursuant to the regulations established by the Chief Librarian of the University of Ottawa. The author also authorizes the University of Ottawa, its successors and assignees, to make reproductions of this copy by photographic means or by photocopying and to lend or sell such reproductions at cost to libraries and to scholars requesting them.

The right to publish the thesis by other means and to sell it to the public is reserved to the author, subject to the regulations of the University of Ottawa governing the publication of theses.

N.B. LE MASCULIN COMPREND ÉGALEMENT LE FÉMININ

Dec. 17, 2002 — DATE

Sudan Wang — (AUTEUR) SIGNATURE (AUTHOR)

Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

WANG, Sudan

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M.Sc. (Systems Science)

GRADE - DEGREE

Faculty of Graduate and Postdoctoral Studies

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Issues in Adding Special Purpose Functions to Free Scientific or Statistical Software

John C. Nash

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

D. Lane

R. Quon

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

SIGNATURE

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

University of Ottawa

School of Management

Master's Program in Systems Science

M.Sc. Thesis

Issues in Adding Special Purpose Functions to Free Scientific or Statistical Software

Student Name: Sudan Wang

Thesis Supervisor: Prof. John C. Nash, D.Phil.

School of Management

University of Ottawa

Date: **Dec. 10, 2002**

Abstract

Free Software offers people the freedom to share or change the software. By accessing the source code, people can adapt or modify the software to meet their special needs. This facilitates research, education and business. However, adding special purpose functions to Free Software is still a big challenge for developers. Though some software packages provide functions that have potential utility for certain needs, these functions might not be directly used or are very difficult to use, especially by students.

This thesis aimed to provide developers with some feasible approaches to assisting novice users. It attempts to add forecasting functionality and easy-to-use user interface to different types of software packages, such as R and Scilab. Several example interfaces are implemented in multiple programming languages (such as Perl, JavaScript and HTML) on the Windows and Linux platforms. We conducted analyses on the attempts made and the lessons learned in these implementations. This provides prototypes and caveats for further efforts, which may have relevance to effectively exploiting Free Software, especially for online teaching.

Acknowledgements

Writing the thesis made a difference to my understanding of statistical analysis and modeling work, programming skills and research style. I am indebted to all people who gave me help, support and encouragement.

I am most grateful to Dr. John C. Nash, my supervisor. Without his guidance, support and encouragement, I would not have been able to complete the thesis so quickly and smoothly. Early in the first meeting last August, he offered me a well-defined research outline and later kept me on track. Whenever I got frustrated along the way, he gave me good advice and encouragement. The methodical and efficient research approaches he taught me will keep benefiting me in the future.

Also, I would like to thank Dr. Jean-Michel Thizy, Dr. Tony Quon and Dr. Daniel E. Lane, whose significant comments and constructive suggestions helped in shaping the thesis.

Special thanks go to Mary M. Nash, who kindly proofread the thesis and gave me valuable suggestions on polishing it.

Many thanks and much love to my parents and husband for being a constant source of inspiration and support in my life.

Table of Contents

List of Figures

Chapter 1 Introduction

Since the 1980s, Free Software has played an important role in promoting the development and distribution of software. It has fundamentally changed the way software can be produced and distributed. According to the definition given by the Free Software Foundation (FSF) (undated-a), Free Software is software that users can run, redistribute, adapt and improve without limit. This means that users may freely access the software and the source code. They can learn and make use of new computer techniques. They can make modifications to meet their special needs. They can also learn and reference programming skills from the wide variety of source code. Meanwhile, the broad user community greatly helps with testing and suggestions for the development of Free Software. Thus, Free Software benefits both users and developers.

As forecasting is one of our main research interests, we would like to take it as an example of users' special needs. Users may want to conduct an analysis via a certain forecasting method, which is not provided by their proprietary software packages in hand. Since proprietary software does not offer users the source code, users cannot make their own modification. In this case, developers can make use of Free Software to meet users' need by modifying the source code. Today more and more people devote their attention to Free Software and take advantage of it. They learn, use or improve Free Software. This facilitates research, education and business.

However, the modification of Free Software is not always easy and convenient. In the above case, developers often encounter some difficulties when adding forecasting functionality to free scientific or statistical software. They may find built-in functions that can be directly used for that forecasting method. Or the user interface provided by the software package is too difficult to be used by novice users. (We note that provision of interfaces to proprietary software may also be useful in this respect.) Sometimes the documentation is not very detailed or systematic. Sometimes it is difficult to find a large number of examples to show how to use the computational resources step by step. These kinds of problems exist widely in free as well as proprietary software.

This thesis aimed to look for feasible and effective approaches to solving the above problems. It attempted to explore the key issues in adding special purpose functions to different types of free scientific or statistical software packages by adding forecasting functionality, creating an easy-to-use user interface and conducting analyses on these attempts.

The statistical software S-Plus and the scientific software Matlab are very popular proprietary software products, especially in educational institutions. R and Scilab respectively are viewed as their counterparts in the Free Software community and have established the wide base of users. We believe that the research based on these packages could be representative. Hence the thesis used R and Scilab as examples and created several interfaces for them. These interfaces were implemented in multiple programming languages, such as Perl, JavaScript and HTML, on the Windows and Linux platforms. These example interfaces may assist novice users to learn forecasting techniques or computational languages. Most importantly, this research attempted to provide programmers with prototypes to build working interfaces to computational tools. The lessons learned in designing and implementing these interfaces could be helpful for further efforts in this field. This may improve computer assistance for online teaching and promote the dissemination of Free Software.

Chapter 2 A Brief Review of Forecasting Methods

This chapter introduces the importance of forecasting, gives one kind of classification of forecasting approaches and reviews three main forecasting tools related to the thesis. The purpose of this chapter is to provide our perspective on forecasting, since tools within software may have facilities with similar names and acronyms that in fact take a very different viewpoint and may not do what we want. In other words, this chapter states our forecasting technique requirements for the end-product seen by the user.

2.1 Need for Forecasting

Since people live in an uncertain world, the future remains more or less obscure. We cannot exactly know whether it will rain tomorrow, or whether the price of a stock will rise next week. This uncertainty makes it difficult for people to make decisions. Should we plan a camping trip tomorrow, or should we invest in a stock next week? In order to make intelligent decisions, people have to predict the future events, that is, forecast.

Besides individuals, organizations of all kinds increasingly depend for their planning on forecasting. The government of a country needs to predict the population growth rate, the unemployment rate and the economic growth rate in order to formulate appropriate policies (Bowerman and O'Connell, 1987). A factory needs to predict the market demand and the price of materials in order to plan its production. High-quality forecasting means a lot to organizations, both in the short and long term.

Forecasting has played an increasingly important role in the activities of governments and other organizations. This is due mainly to the rapid development of society. The economy, politics and technology evolve rapidly. Meanwhile, the structure of society and the relationships among elements in the society become more complex. Moreover, organizations have to react quickly to the changing environment. This should encourage organizations to view forecasting as a very important activity.

Fortunately, computers have provided critical facilities for improving the quality of forecasting. The rapid development and the complexities just discussed imply the need to manipulate large amounts of data. Computers make advanced quantitative techniques

possible and efficient. This not only expands the application scope of forecasting, but should also increase the accuracy and reliability of forecasting.

2.2 Approaches to Forecasting

Existing forecasting approaches are generally grouped as being either quantitative or qualitative. Quantitative approaches are based on the manipulation of mainly numerical data. In contrast, qualitative approaches involve the use of forecasters' judgment or opinion (Hanke and Reitsh, 1998). Both approaches are widely used, and good forecasting involves both of them.

We can classify quantitative approaches as time series techniques or causal techniques.

A time series is a set of observations, indexed by time (Newbold and Bos, 1990). The time series techniques assume that certain patterns exist in the observations and that the past patterns will repeat in the future (Makridakis and Wheelwright, 1987). In other words, the data patterns will be preserved with the evolution of time. The time series techniques include naïve or *ad hoc* methods, exponential smoothing, decomposition, ARIMA and other methods. In the next section, three major techniques will be further explained.

In contrast, causal techniques are based on the relationships among various factors (Makridakis and Wheelwright, 1987) and assume that these relationships will not change in the future. These techniques attempt to seek explanatory factors and discover how the factors interact with each other. The behavior of factors will be projected into the future to develop predictions. Regression methods are the most common examples.

In some situations, qualitative approaches must be relied on to make forecasts. For example, there might be little or no relevant historical data that are available. Then the use of good judgment might become essential to the forecasting. Sometimes these techniques are called judgmental forecasting. Delphi, growth curves and scenario writing are some of popular qualitative methods, though it is clear that growth curves are extrapolated using numerical computations.

We present a simple graph (Figure 2.1) to illustrate the above classification, which is based on an idea of Newbold and Bos (1990, Chart 1-2 "A Taxonomy of Forecasting Approaches/Methods").

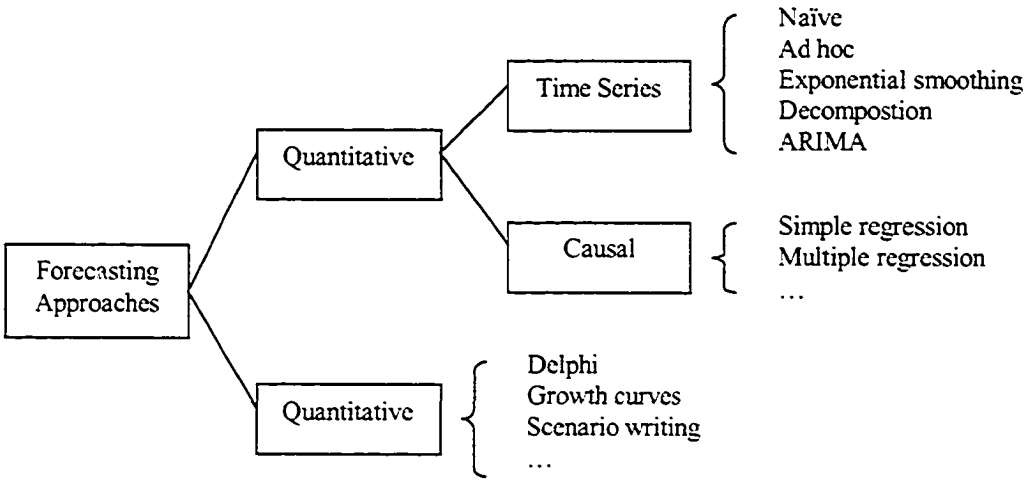

Figure 2.1. Classification of approaches to forecasting

2.3 Some Important Forecasting Tools

We begin with some basic notation and terminology.

1. Basic notation

- t: the time index, used to identify a time point or time period during which measurements are made.
- Y(t): the actual value of a quantity of interest Y at time point t (or time period t); the sequence Y(1), Y(2), ... makes up the time series.
- F(t): the forecasted value of Y(t).
- e(t)=Y(t)-F(t): the residual, sometimes referred to as the "error".
- S(t): the value of the smoothed series at period t (or called smoother).
- AR: the autoregressive parameters.
- SAR: the seasonal autoregressive parameters.
- MA: the moving average parameters.
- SMA: the seasonal moving average parameters.
- B: the back shift or lag operator such that B Y(t)=Y(t-1) and Bm Y(t)=Y(t-m).

2. Terminology

- Time series: A time series is a sequence of measurements of some numerical quantity made at or during successive periods of time (Farnum and Stanton, 1989).
- Model: A model is a mathematical description of the process which generates a given time series (Farnum and Stanton, 1989).
- Sample autocorrelation function (ACF): The sample autocorrelation function refers to a function of sample autocorrelation at lag k (k=1,2, ...) (Bowerman and O'Connell, 1987). Consider a (original or transformed) time series Y(b), Y(b+1),..., Y(n). The sample autocorrelation at lag k, denoted by r(k), is

$$r(k)=\frac{\sum_{t=b}^{n-k}(Y(t)-\bar{Y})(Y(t+k)-\bar{Y})}{\sum_{t=b}^{n-k}(Y(t)-\bar{Y})^2}$$

where,

$$\bar{Y}=\frac{\sum_{t=b}^{n}Y(t)}{n-b-1}$$

- Sample partial autocorrelation function (PACF): Similar to sample autocorrelation r(k), the sample partial autocorrelation r(k,k) at lag k (Bowerman and O'Connell, 1987) is

$$r(k,k)=\begin{cases} r(1) & \text{if k=1} \\ \dfrac{r(k)-\sum_{j=1}^{k-1}r(k-1,j)r(k-j)}{1-\sum_{j=1}^{k-1}r(k-1,j)r(j)} & \text{if k=2,3,...} \end{cases}$$

where,

$$r(k,j)=r(k-1,j)-r(k,k)r(k-1,k-j) \qquad \text{for j=1,2,...,k-1}$$

2.3.1 Winters' Smoothing Method

"Exponential smoothing is a forecasting technique that attempts to 'track' changes in a time series by using newly observed time series values to 'update' the estimates of the parameters

describing the time series" (Bowerman and O'Connell, 1987). The approach weights each of the observed time series values unequally, with more recent observations being weighed more heavily than more remote observations. This unequal weighting is achieved through one or more smoothing constants, which determine how much weight is given to each observation.

Winters' smoothing method is one of exponential smoothing methods. Since it is based on Holt's smoothing method, some people also call it Holt-Winters' smoothing method. It can be used to forecast time series that show trend and seasonality. We smooth the level of the series by the formula (Nash and Nash, 2001):

$$S(t) = \alpha * Y(t)/I(t-L) + (1-\alpha) * S(t-1) + b(t-1)$$

where:

S(t) is the value of the smoothed series at period t (or called smoother);

Y(t) is the actual value at period t;

L is the length of seasonality, that is, the number of periods in a cycle;

b(t) is the trend smoother and $b(t) = \gamma * (S(t) - S(t-1)) + (1-\gamma) * b(t-1)$;

I(t) is the seasonal smoother and $I(t) = \beta * Y(t)/S(t) + (1-\beta) * I(t-L)$.

Here, α, β and γ are the smoothing constants, with values between 0 and 1. Thus, the k-period ahead forecast is calculated:

$$F(t+k) = (S(t) + k * b(t)) * I(t-L+k)$$

Note that notation may differ in different books. Some books use β as the smoothing constant for the trend smoother and γ for the seasonal smoother (see Hanke and Reitsch, 1998).

2.3.2 Classical Time Series Decomposition

This approach assumes that every time series is composed of a number of component parts. The classically defined component parts are trend (T), cycle (C), seasonal variations (S), and irregular fluctuations (I).

Trend (T) refers to the upward or downward movement of a time series over a period of time (Bowerman and O'Connell, 1987). Thus, it reflects the long-run growth or decline in the time series.

Cycle (C) refers to recurring up and down movements around trend levels (Bowerman and O'Connell, 1987). These fluctuations take place over a period of years. The length of time measured from peak to peak or from trough to trough may be not fixed.

Seasonal variations (S) are periodic patterns in a time series within a calendar year (Bowerman and O'Connell, 1987). Such fluctuations repeat year after year. Seasonal variations are usually caused by factors such as weather and social customs.

Irregular fluctuations (I), or random errors, represent the left-over movements in a time series after trend, cycle, and seasonal variations have been accounted for (Bowerman and O'Connell, 1987). Such movements are erratic movements that follow no regular pattern.

The four components may be independent of all of the others, also called additively related; thus the behaviour of the series is simply the sum of its parts. Or the components may be not completely independent of one another, that is, multiplicatively related.

Additive model: $Y = T + C + S + I$

Multiplicative model: $Y = T * C * S * I$

The classical techniques used to decompose time series is the "ratio to moving average" which identifies both seasonality and cyclical behaviour so that the former may be removed and the latter isolated for other purposes. The method assumes a multiplicative model:

$$Y(t) = T(t) * C(t) * S(t) * I(t)$$

The following explanation is based on the ideas of Nash and Nash (2001, chapter 14).

The first step is to compute centred moving averages with the length of seasonality, L. The moving averages, donated by M(t), contain trend and cycle, because seasonality and error should be eliminated or "averaged out".

$$M(t) = T(t) * C(t)$$

Secondly, divide the time series by the moving averages and get the ratio of actual-to-moving averages.

$$\frac{Y(t)}{M(t)} = \frac{T(t) * C(t) * S(t) * I(t)}{T(t) * C(t)} = S(t) * I(t)$$

Then, average the resulting values corresponding to the same season and obtain the seasonal factors by separating errors.

Next, divide the time series by the appropriate seasonal factors and get the deseasonalized series, denoted by D(t).

$$D(t)=\frac{Y(t)}{S(t)}=\frac{T(t)*C(t)*S(t)*I(t)}{S(t)}=T(t)*C(t)*I(t)$$

From the deseasonalized series D(t), compute the trend line T(t) via simple linear regression.

Next step: divide the deseasonalized series D(t) by the trend T(t) to get the preliminary irregular series.

$$\frac{D(t)}{T(t)}=C(t)*I(t)=\frac{Y(t)}{T(t)*S(t)}$$

Thus, the forecasting model with C(t) is

$$F(t)=T(t)*S(t)*C(t)$$

Figure 2.2 presents the basic steps employed in the decomposition procedure, which is based on the ideas of Nash and Nash (2001, p184-186).

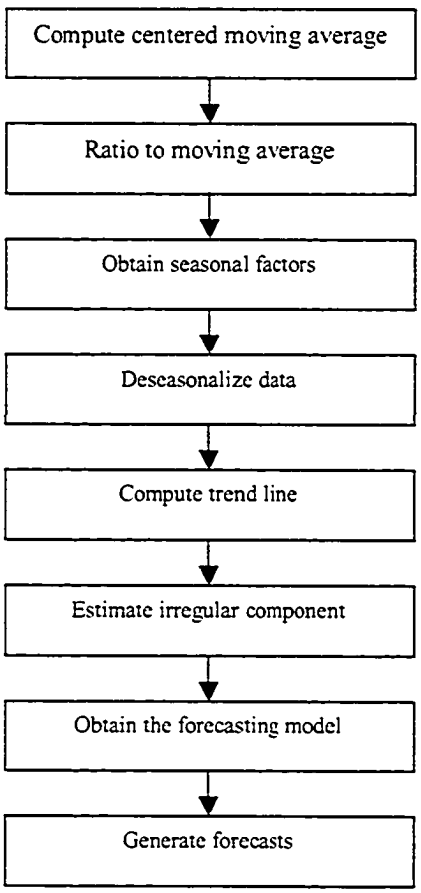

Figure 2.2. The procedure of classical time series decomposition (based on the ideas of Nash and Nash [2001, p184-186]).

2.3.3 Simple ARIMA Modeling (via the Box-Jenkins Approach)

ARIMA stands for Auto-Regressive Integrated Moving Average. Following the suggestion of John C. Nash, I took the notation used by Minitab and other software to work with ARIMA models.

In general, a simple ARIMA model is specified by means of an expression of the form:

ARIMA p d q P D Q L

Where,

the parameters p, d, q define the non-seasonal part:

p is the number of autoregressive parameters;

d is the degree of differencing;

q is the number of moving average parameters;

the parameters P, D, Q, L define the seasonal part:

P is the number of autoregressive parameters;

D is the degree of differencing;

Q is the number of moving average parameters;

L is the length of one season.

It is initially written down (Nash and Nash, 2001):

$$(1-\sum_{i=1}^{P} SAR_{iL}B^{i*L})(1-B^{L})^{D}(1-\sum_{j=1}^{p} AR_{j}B^{j})(1-B)^{d}Y(t)$$

$$=\mu+(1-\sum_{m=1}^{Q} SMA_{mL}B^{m*L})(1-\sum_{k=1}^{q} MA_{k}B^{k})e(t)$$

The Box-Jenkins technique, developed by George Box and Gwilym Jenkins, relies heavily on the use of three familiar time series tools: differencing, the autocorrelation function (ACF), and the partial autocorrelation function (PACF) (Box and Jenkins, 1976). The basic stages in the Box-Jenkins procedure include tentative identification, estimation, diagnostic checking, and forecasting. Figure 2.3, based on Hossein Arsham's chart (Arsham, 1996), shows the iterative procedure. Firstly, if necessary, we make a time series stationary by differencing it and compute an ACF and PACF to tentatively identify a model. Then we estimate the model parameters. After that, we use diagnostic tests to check whether the model is adequate. If the model proves to be inadequate, we return to the step of identification and improve the model. Once a final model is determined, we use it to forecast future time series values.

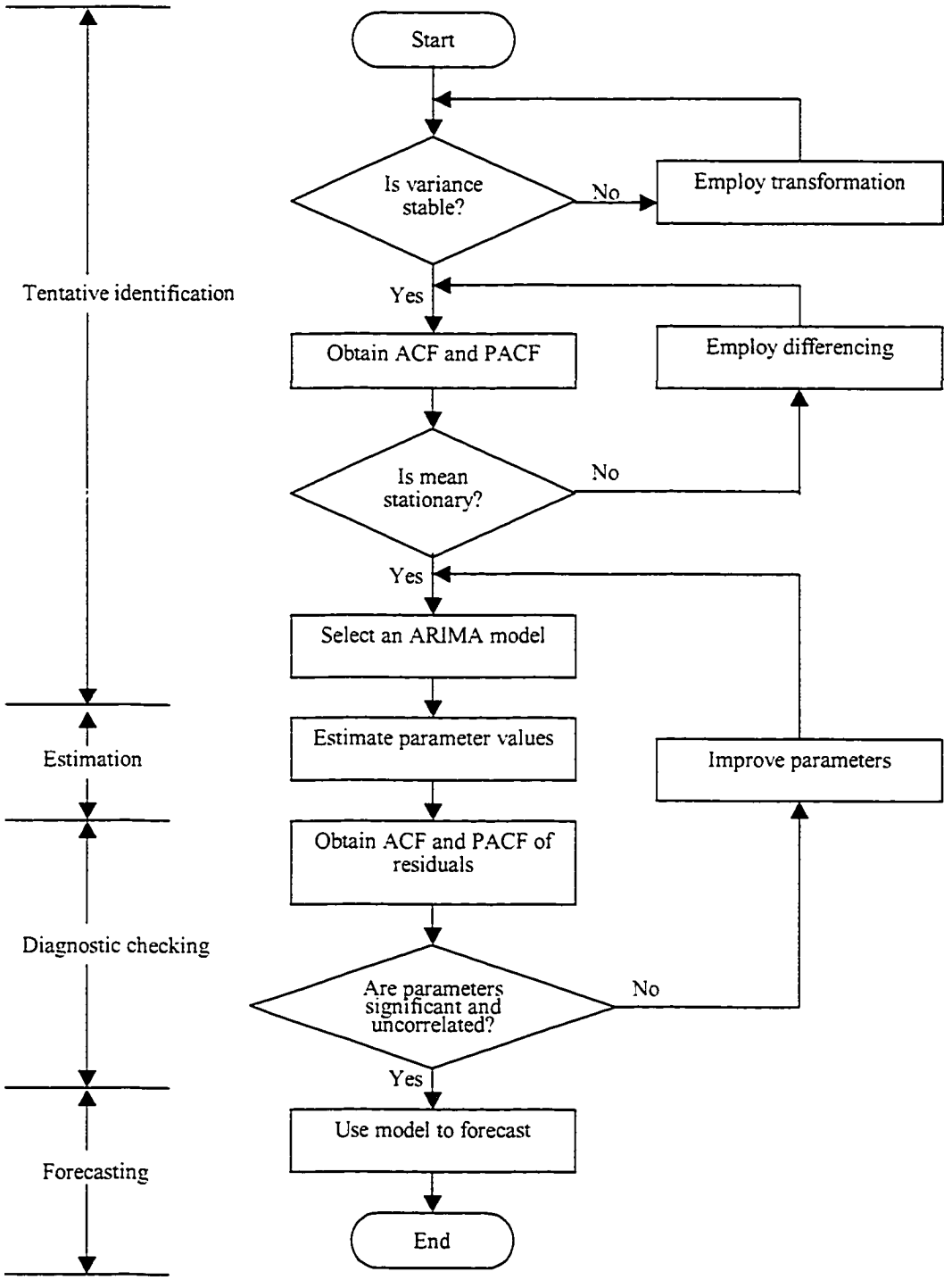

Figure 2.3. Simple ARIMA modeling (via Box-Jenkins method) flowchart (based on Hossein Arsham's chart)

Chapter 3 The Research Question

This chapter explains the classification and advantages of Free Software, and introduces two free scientific or statistical packages, R and Scilab. Then the difficulty in using the two packages is presented to show how the research question arises and why it is worthwhile to be discussed. These are used to justify the research.

3.1 Free Software

In the community that advocates freely distributing software in source code format, there are many discussions on choosing "Free Software" or "open source software" as the term used to refer to their software. The above terms refer to more or less the same thing. Bruce Perens of the Open Source Initiative explained this situation as follows:

> "*Free Software* as a *political* idea has been popularized by Richard Stallman since 1984, when he formed the *Free Software Foundation* and its *GNU Project*. Stallman's premise is that people should have more freedom, and should appreciate their freedom. He designed a set of rights that he felt all users should have, and codified them in the *GNU General Public License* or *GPL*. Stallman punningly christened his license *the Copyleft* because it leaves the right to copy in place. Stallman himself developed seminal works of Free Software such as the *GNU C Compiler*, and *GNU Emacs*, an editor so alluring to some that it is spoken of as if it were a religion. His work inspired many others to contribute Free Software under the GPL. Although it is not promoted with the same libertarian fervor, the Open Source Definition includes many of Stallman's ideas, and can be considered a derivative of his work."(Perens, undated)

In this thesis, the term "Free Software" defined by Free Software Foundation (FSF) is chosen and some related terms such as commercial software and proprietary software will be briefly explained.

3.1.1 Classification of Software and Related Terms

The following classification is based on the categories and the definitions of the FSF.

According to whether users can freely own a copy of software, we can classify all software in two categories: Free Software and proprietary software (Figure3.1). According to the FSF's definition, Free Software is software that users can run, redistribute, adapt and modify without limit. By contrast, proprietary software is software whose use, redistribution or modification is prohibited, or requires users to ask for permission.

There is some confusion between proprietary software and commercial software. They are not the same thing. Commercial software is software whose developer aims to make money by selling it. Some commercial software products are proprietary, but others are Free Software.

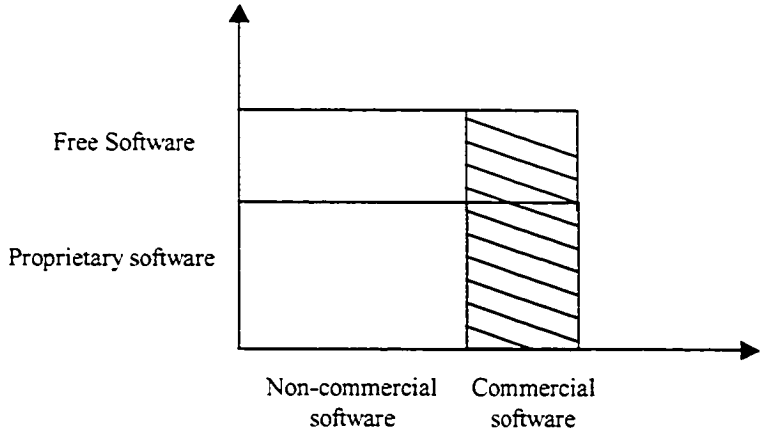

Figure 3.1. The classification of software (based on Categories of Free and Non-Free Software by FSF)

3.1.2 Advantages of Free Software

According to the definition given by the FSF, Free Software is free of intellectual property claims. The word "free" refers to freedom, not to price. The freedom for the users of a program includes freedom to run, redistribute, adapt and improve the program without limit. Free Software brings four main advantages to society as a whole.

1. Users may usually access Free Software without payment.

Any user, person or organization, can usually install and use Free Software for any purpose without the need to ask or pay for permission (FSF, The Free Software Definition). Users are also free to redistribute copies, even sell copies, as long as the source code is accessible. To get copies of Free Software by mechanisms such as FTP and CD-ROM, people may have to pay, although they can download most Free Software from web sites without charge. Though time consuming, it does not cost anything beyond connection charges. To obtain large amounts of software by CD-ROM, people may generally only pay for the physical disk and shipping. This manner of distribution allows wide access to Free Software.

As for proprietary software, people have always had to pay for the use of a package, whether or not it meets their needs. Without payment, they do not have use of the package. But even if they do pay, it may turn out to be unsuitable, yet many licensers refuse refunds. One can argue that the total contribution of a package to all of society has increased by making it "free".

2. Users can access the source code of Free Software.

Having a copy of the source code, users can make necessary changes and adapt the program to their own needs. This means they can use the source code for reference and base further work on it. Thus, on the basis of an existing program, one user can rewrite parts to add a feature and then another one may continue to rewrite parts to add yet other new features. The evolutionary approach makes software development more efficient and productive. This is the most important advantage of Free Software.

Users of proprietary software cannot do this. Usually they are not allowed to have copies of the source code. What they actually own is only the executable files or possibly only the right to run them on particular machines even after they buy the files. They cannot add favorite features or do any modification. Even if users just want to fix small problems or make little improvements, they must wait until the developer publishes a patch or they have to re-program from scratch. This inevitably results in duplicate development effort.

3. Free Software benefits education and training.

On the one hand, browsing, studying and modifying a variety of source code is a good way for students and novices to learn useful programming skills. But generally they cannot see the source code of proprietary software. Thanks to Free Software, they can access a vast amount of source code, examine all kinds of programming styles and study intensively the sophisticated skills of writing codes. They may become mature developers more quickly.

On the other hand, most Free Software packages are zero-cost, which can improve and promote computer assistance for teaching. Due to the high price of commercial software, many students and educational institutions cannot afford it. This is true especially for those in developing regions and countries. Table 1 presents the current price of several popular scientific or statistical software packages. Most of the prices are for the base versions. It is easy to see that these software packages are very expensive. Though cheaper, some academic versions do not cover full features or have a limitation of use time.

Table 1: The price of several popular scientific and statistical software packages (in SUS)

Software / Type	MATLAB	SAS	S-Plus	Minitab
Commercial version	$1900 (version 6.5)	$30-$120[a] (version 8.2)	$1779.95[b]	$1195 (version 13)
Academic version	$99 (version 6)	$125 (learning Edition 1.0)	Free (one year limited)	$99.99 (version 13)

[a]: the annual price

[b]: from the price list of Dynacomp Software

Thanks to Free Software and the Internet, advanced techniques and useful information can be freely accessed with little or no payment. Today many educational institutions take full advantage of Free Software, even in developed countries. There are some universities that use R as their lab software. As an example, the web site of the Department of Statistics & Actuarial Science, University of Waterloo, Canada writes:

"At Waterloo, we encourage users to use R, as there are no restrictions on the number of users or installations. With Splus, a maximum of 50 simultaneous users at UW are permitted."

R seems more welcome for professors and students at University of Waterloo.

4. Free Software also benefits developers.

Making a software project free is an effective solution to growing the user base. Today more and more developers of software realize that a broader user community greatly helps with testing and suggesting improvements. This makes the software testing and refinement process faster and better. Benefiting from all kinds of suggestions from users, the developers can develop the project to its full potential. It also means higher-quality software for users.

5. Free Software permits users to have a stable software base.

Software vendors of proprietary products may impose requirements on users that are inconvenient or costly. Forced to accept an "upgraded" package, users often face such inconvenience as rising price and changing features. Sometimes add-on materials no longer work. This makes users feel restrained. On the contrary, Free Software offers users the freedom to share or change the software. Users can use or distribute Free Software as long as possible. There is no worry about expiration of the use. If necessary, they have the source code to work with. Free Software gives them a measure of control.

Now let us examine an example of how Free Software affects society. Linux may be the most prominent example among Free Software. Linux is an operating system whose kernel was written by Linus Torvalds, a student of University of Helsinki in Finland (Linux online, undated). It is developed under the GNU General Public License and the source code is freely available. Of course, the distribution may be not free. Some companies and developers may charge money for them. Due to the features of free availability, strong functionality and outstanding reliability, Linux has become quite popular and a vast number

of programmers have taken the Linux source code and adapted it to meet their individual needs (L

inux online, undated). The success of Linux encourages more people to become involved in Free Software.

3.1.3 Free Scientific and Statistical Software

There exist hundreds of free scientific and statistical software packages. R and Scilab are two powerful and popular packages. They share a heritage with the proprietary packages S-Plus (for R) and Matlab (for Scilab), so there is a wide, established base of users. Given their wide applicability to many problem classes, this thesis takes these two packages as examples upon which to base the research.

3.1.3.1 R

R, a GNU project for statistical computing and graphics, is available in source code form. It was initially created by Robert Gentleman and Ross Ihaka of the University of Auckland, New Zealand. Its language and environment is similar to that of S that was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues (Venables and Ripley, 1994). It provides a wide variety of statistical methods (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering) and graphical techniques. R compiles and runs on a wide variety of platforms including UNIX, Linux, Windows and MacOS.

The R environment is a system that provides high-level graphics, interfaces to other languages and debugging facilities. Its facilities include data manipulation, calculation and graphical display. R can be easily extended via add-in packages. There are many such packages that cover a very wide range of modern statistics.

The R language is considered a dialect of S. It is a functional programming language with control-flow constructions. It allows users to add new functionality by defining new functions and calling C, C++ and FORTRAN code at run time.

3.1.3.2 Scilab

Scilab, developed at the French National Institute for Research in Computer Science and Automatic Control (INRIA), is an open source (but not strictly Free Software under the FSF definition) scientific software package for numerical computations. It is also considered a powerful package for system control and signal processing applications. It has a user-friendly environment and many features similar to MATLAB. MATLAB is not distributed free.

Scilab consists of three main parts: an interpreter, libraries of functions and libraries of FORTRAN and C routines. (see Scilab home page, http://www-rocq.inria.fr/scilab/). It includes the environments for open programming, modeling and simulation of dynamical systems and network analysis.

Scilab provides a high-level programming language. The syntax is similar to that of Matlab. A key feature is that it allows users to manipulate numerical matrices and complex objects such as rational or polynomial transfer matrices. It is easy to create functions and libraries of functions. For example, functions are treated as data objects and can be passed as input or output arguments of other functions. Scilab can be interfaced with C or FORTRAN programs, allowing users to access the libraries of Scilab.

3.2 The Research Question

Some proprietary scientific or statistical software packages provide easy-to-use user interfaces for time series modeling. However, they may only provide a limited number of forecasting methods. For example, users of S-Plus cannot conduct forecasting analyses using the Time Series Decomposition (TSD) method on the menu-based interface. Knowledgeable users can write functions for TSD using the S language, and they could even provide an interface to S-Plus for this purpose. However, the next version of S-Plus could render this inoperative. In such situations, Free Software (in this case R), allows us to maintain our capability. Moreover, if absolutely necessary, we can work with the source code.

However, the task of adding special purpose functions to free scientific or statistical software is not always easy. These special functions could be forecasting, optimization or

other such tasks. To complete such tasks using R or Scilab as the basis, developers may encounter the following difficulties, which we illustrate with reference to forecasting.

- *R or Scilab lack functions for traditional forecasting.*

There is a plug-in package for time series in R. The *ts* (time series) package has a number of functions for modeling and plotting time series. It defines a type of object for time series. Time-series objects, like other objects such as function, can be manipulated as data. The package provides the ARIMA method, the Holt-Winters' method and the classical decomposition method for fitting time series. Since R is not software specifically intended for forecasting, it does not provide all prediction functions for the above three methods. For example, the package has the function *decompose( )* which computes time series decomposition filtering for a given time series, but lacks a corresponding function for forecasting. Users cannot conduct a forecasting task via the time series decomposition method using built-in functions. Since the decomposition method is one of the main forecasting tools in general use, forecasters will find R inconvenient without this facility.

As for Scilab, it has no function for Winters' smoothing or classical time series decomposition. It does have an *ARMA* library for system identification, but it aims to simulate an ARMAX (Auto Regression Moving Average with eXogenous variables) process rather than to forecast (Scilab Group, undated).

- *There is not an easy-to-use interface.*

The current user interface of R or Scilab can only provide rather simple menu choices such as loading source files, loading packages and saving log files. Most manipulations have to be completed with command lines. For proficient programmers, this should not be very difficult. For novices, however, it is very difficult and frustrating, especially if they do not always know what commands they should use. To make forecasts, a user may need to load the observed data, convert the data to a time-series object, choose parameters for modeling and plotting functions, save corresponding outputs, and feed these back into the prediction tools. Without a graphic interface, these operations always confuse novices.

- *The documentation is not very detailed or systematic.*

There are three kinds of help systems in R or Scilab. One help system for searching information on functions can be invoked from the command line. One help system for manuals can be accessed from the Help item on the main menu. The other help system for mailing list archives can be accessed from the respective web sites. Though the above systems do provide good help facilities, it is still a difficult and time-consuming job for developers to look for information that should be "obvious". For example, developers wanting to know about data types defined in R have to read manuals or search in mailing list archives. Very often they cannot find exact information. The author faced this kind of situation when learning the basic manipulations of R. It was frustrating and tedious.

- *It is difficult to find a large number of examples of usage.*

Except for the help files and the manuals edited by the development teams, there are only a limited number of tutorials written by other people. Developers cannot find enough examples when they learn how to use functions. When the author learned how to invoke R from the Windows command line, she could only find one simple example. But she still could not discover how to set the parameters. This was resolved by working with her supervisor, Dr. John Nash, over several weeks.

Though some publications about S may be helpful, R is not exactly S. They have distinct differences in some aspects. As far as scoping is concerned, "whereas S (like C) by default uses *static* scoping, R (like Scheme) has adopted lexical scoping" (Hornik, 2002). As for the modeling code, there are some differences to which users may have to pay attention. Here is a simple example (Hornik, 2002) to regress y on x^3. In S the command is lm(y~x^3); while in R, the command is lm(y~I(x^3)). In addition, there are actually some differences resulted from incorrect coding in R.

This is also true for Scilab. R and Scilab are not the only software packages that have the above disadvantages. Many or even most Free Software products have similar problems. We aimed to look for effective approaches to help developers to solve these problems.

3.3 Objectives of Implementations

To find feasible approaches to solve the above problems, the author designed and implemented several example interfaces for R or Scilab. These interfaces aimed to provide the following general functionalities.

- *The core computational and graphical tools for forecasting with time series models based on R or Scilab.*

Specifically speaking, we want to allow users to do the following work:

  - Loading data from external text files;
  - Fitting the data via such methods as Holt-Winters' smoothing, time series decomposition or ARIMA modeling;
  - Making forecast on the basis of the above models;
  - Outputting computation results and graphics.

- *An easy-to-use graphical interface to the tools in R or Scilab.*

The interfaces assume that users have some familiarity with time series models, but require no knowledge of the R or Scilab language.

The interface can facilitate and simplify the operations for forecasting. With the aid of the interface, users may perform forecasting easily and efficiently without any worry about how model-fitting or forecasting functions are implemented. What they need to do is to check radio buttons or checkboxes, type several numbers in textboxes and click a button to invoke an operation. They can choose a data file as input either from a floppy disk or a hard disk. They can enter relevant parameters of models such as model type and smoothing weight. They can also choose favourite data and/or graphics as output.

- *Didactic support to the interface.*

The interface can also provide supporting information to explain the methods, tools, or motivations. When users need to make choices among checkboxes or radio buttons, they may get corresponding supporting information. When they want to know about forecasting

techniques, they may find some explanations. By doing so, the interface is friendly and helpful for users.

3.4 Significance of the Research

This research should have relevance to users who wish to forecast, to educators and students of forecasting and to the community associated with Free Software.

- *The example interfaces assist novice users to learn forecasting techniques or new computational languages.*

The graphical user interfaces simplify the use of Free Software for forecasting by providing supporting information and other directive mechanisms such as log files. This may be helpful for novice users. Providing supporting information makes setting model parameters easier. Users only need to follow conductions shown on the interfaces. For some people who wish to learn new computational languages, log files can be beneficial. By reading log files, they can learn how to write forecasting functions or generate text or graphical outputs step by step. In this way, users may become familiar with manipulations and languages readily and quickly. They may get encouragement to keep learning.

- *The thesis provides prototypes or models showing how to adapt and modify Free Software.*

We illustrate how to design and implement working interfaces to free computational tools for forecasting tools. We recorded the attempts made and conducted analyses on the lessons learned. Several prototypes or models of adding special purpose functions to Free Software are provided. The solutions and suggestions may help other people for continuing efforts in this field.

- *The examples show how to provide special features useful in online teaching.*

Online teaching allows disadvantaged students to establish connections to the rest of the world and get opportunities to study advanced scientific techniques. It plays an increasingly important role in education in both developed and developing areas of the world.

By building our examples on Free Software we indirectly promote its use. This may provide useful solutions and references for online teaching and/or the use of computation in forecasting and related courses. More generally, the approaches suggested here can be applied to other tasks for improving and adapting Free Software.

Chapter 4 A Web-based Implementation: RForecast

In order to find feasible and effective approaches to answering the research question, the author implemented several example interfaces for R and Scilab. This chapter describes the design and implementation for the RForecast interface, which is used to conduct forecasting analyses using R as basis.

4.1 Analysis and Design

As discussed in Section 3.3, the example interface aimed to provide users with three functionalities:

- The core computational and graphical tools for forecasting with time series models;
- An easy-to-use graphical interface to the tools;
- Didactic support to the interface.

R has powerful and flexible statistical and graphical functions, so we can implement the computational and graphical tools using the R language. Aside from the R built-in functions, we need to write some functions, such as the prediction function for the Time Series Decomposition method, in the R language.

As for the graphical user interface (GUI), we may choose either a local or web-based GUI. Since the thesis attempted to provide solutions and references for online teaching, it would better if the interface were web-based. Users can browse and conduct forecasting analyses using a web browser via the Internet. They do not need to install the interface locally on their own computers, a great convenience.

R can be invoked in batch mode, which we can use as a mechanism to interact between the web server and R. To make the interface easy-to-implement and widely applicable, a simple and effective interactivity mechanism, the Common Gateway Interface (CGI) is chosen.

CGI is a programming interface to communicate between a web server and back-end applications. It is a standard method or convention to process data passed from the client (that is, the user's web browser) to the server using Hypertext Transfer Protocol (HTTP)

(The Computer Technology Documentation Project, undated). This standard is supported by almost all browsers (Shiran, Yehuda and Tomer 1998).

In this case, the CGI programs provide for the following functions:

- *They process data entered by users.* When users enter parameters related to loading data, fitting data or making forecasts, and then click a *Go* button, the web browser will send the data to the web server. The web server then communicates with the CGI programs to process the data received from the client.
- *They call functions in the R package.* CGI programs copy the content of the data file to a text file on the server and create corresponding R command files that call functions in the R package. R will be invoked in batch mode and R command files will be run on the server.
- *They convert one graphic format to another.* Running R command files will output data results in text files and graphics in graphical files. By default, the graphics are in Postscript format. CGI programs will convert Postscript format to GIF format via Netpbm, a graphical processing toolkit. Graphics in GIF format can be processed by most popular browsers.
- *They send outputs to the server.* CGI programs send different types of output to the server in order that users can get them in the user interface.

Figure 4.1 (based on Figure 1.3a, Brenner and Aoki 1996) shows the interactivity mechanism of the system in RForecast.

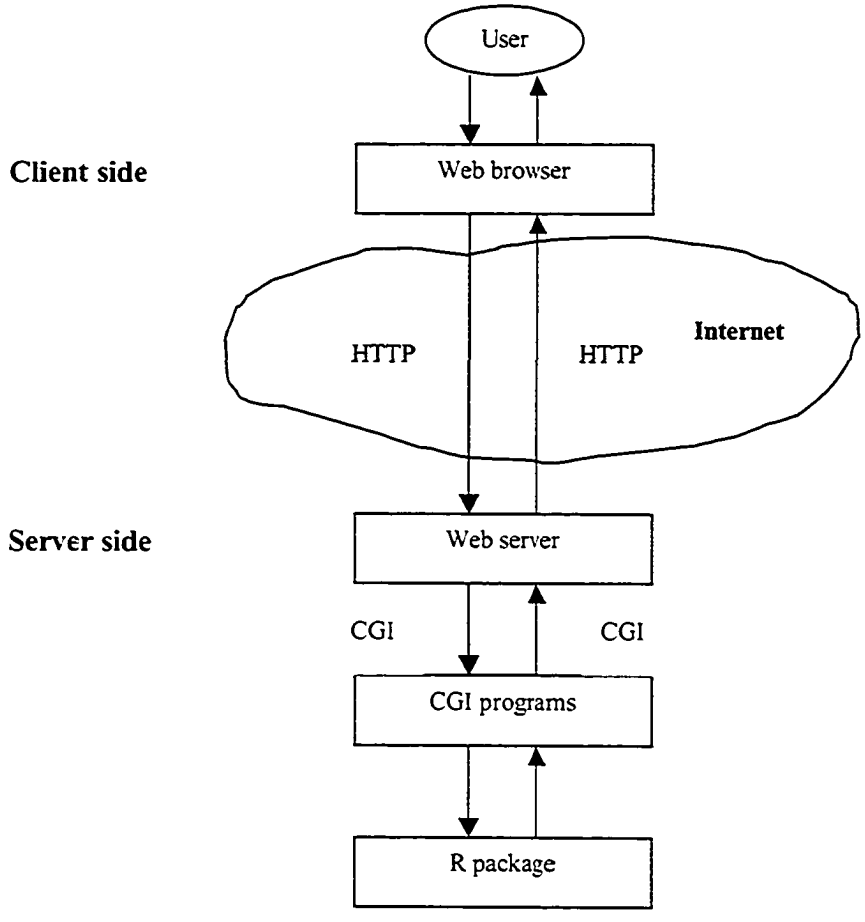

Figure 4.1. The interactivity mechanism of the systems in RForecast

In the system, different types of data flow back and forth between the client and the server. Figure 4.2 illustrates how the data flow through the system. The circles with numbers inside show the time sequence of data flows.

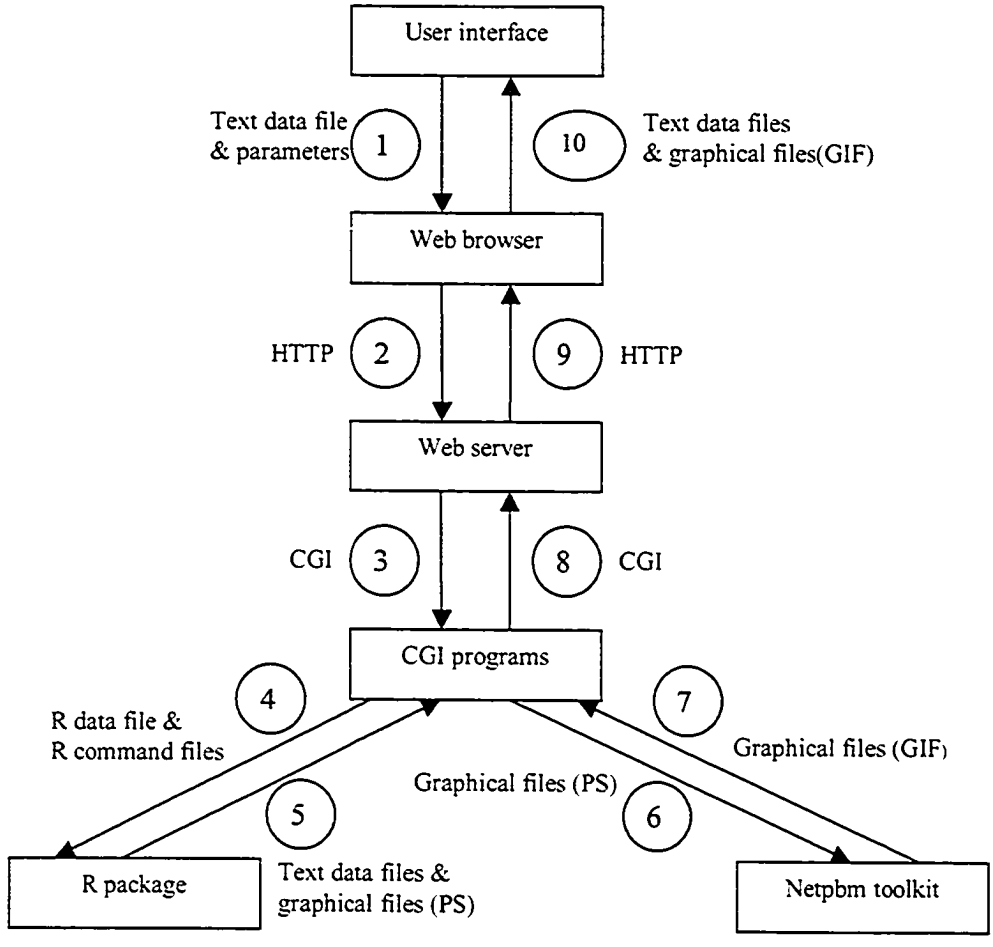

Figure 4.2. The data flows in RForecast (the circles with numbers inside show the time sequence of data flows)

During each call to RForecast, four types of files may be created.

First, CGI programs create a text file on the server if the user chooses a data file from his/her local computer. When CGI programs receive the name of the data file, the file type will be validated. If it is the text type, then the content of the file will be copied to a new text file on the server. The new text file will be used later when running the R command file later. If the user does not choose a local data file, then the sample data file *beer.dat* on the server will be used. In this case no new text file will be generated.

Second, CGI programs create an R command file that is used to invoke R in batch mode. Depending on the parameters entered by the user, the file also includes

corresponding R commands that are used to do some of the following tasks: loading data from a data file, transferring the data to a time series object, model fitting, forecasting and output.

Third, if the user wants to output results, then the R command file will include R commands that are used to output data results or graphical results. After the R command file is run, text files or graphical files will be created on the server.

Finally, if there are graphical outputs, CGI programs will create some temporary files in order to convert these graphics from one format to another.

Here, there are two issues that developers need to note. First, the temporary working files for each call should be removed whenever possible in order to save disk space on the server. Temporary graphics may occupy much disk space. In RForecast, we provide users the choice to delete temporary files (see Appendices B.3 and D.3). The second issue is to identify temporary files to ensure users get their proper outputs. Temporary working files generated for multiple users should be kept separate. Otherwise, theses files may be overwritten if more than one user is calling the RForecast interface at the same time. To recognize temporary files, an identifier should be added to the name of a file. There are several alternatives to set the identifier. Fortunately, Perl has a built-in variable *$$* that retrieves the "process id" for the current CGI script. So RForecast uses "process id" as the identifier for temporary files (see Appendices B.4 line 6 and 327).

4.2 Development Environment and Languages

1. Development Environment

To take full advantage of Free Software, RForecast is developed on the basis of a number of Free Software packages as follows:

The operating system: Linux (Red Hat 7.2);

The R package: R 1.5.1;

The web server: Apache 1.3.22;

Perl interpreter: Perl 5.6.0;

Graphical toolkit: Ghostscript 6.5.1 and Netpbm 9.18.

All the above software packages, installed on the server, are freely available. This shows that it is easy to establish a web-based development environment with the advantage of Free Software.

Though users are recommended to install the R package locally on their own computers, the minimum requirement for them is actually a JavaScript-capable browser. Thus it is clearly feasible to establish an online-teaching environment with the advantage of Free Software.

2. Development Languages

There are three programming or statistical languages used to implement the RForecast interface (see Figure 4.3).

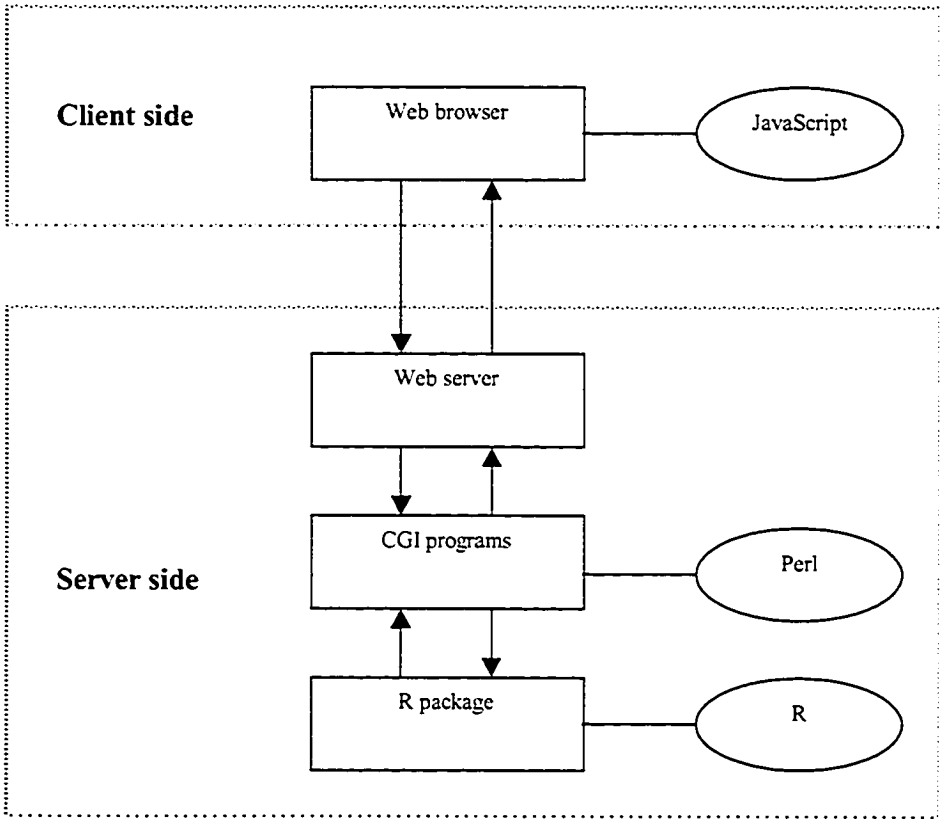

Figure 4.3. The client/server connection and development languages

First of all, the forecasting functionalities are implemented with the R language. R command files are run on the server and corresponding R functions are called on the R environment. These functions include built-in functions and newly defined functions.

Secondly, CGI scripts are written in the Perl language.

"Perl is a script programming language that is similar in syntax to the **C** language and that includes a number of popular Unix facilities such as SED, awk, and tr. Perl is an interpreted language that can optionally be compiled before execution into either C code or cross-platform bytecode. Perl is regarded as a good choice for developing common gateway interface (CGI) programs because it has good text manipulation facilities (although it also handles binary files). It was invented by Larry Wall."(www. searchDatabase.com)

Generally, CGI programs are developed by C, C++, Perl, Unix shells or other languages. Here, Perl is chosen mainly because it is free and cross-platform. In addition, it is relatively straightforward to build and test Perl scripts.

Finally, the JavaScript language is chosen to create client-side scripts in order to reduce the response time between server and client sides.

"JavaScript is an easy-to-use object scripting language designed for creating live online applications that link together objects and resources." (Shiran, 1998) It is supported by most popular browsers such as Microsoft Internet Explorer (IE) and Netscape Navigator (NS), although there are differences in exactly how it is supported.

In RForecast, JavaScript code provides for determining the type of web browser before loading web pages in order to present suitable pages and processing data on the client side before submission to the server.

In this case, R commands and JavaScript scripts are actually embedded into CGI programs written in Perl. In other words, CGI programs will be run to generate HTML pages with JavaScript scripts for the client side and create R command files for the server side. Furthermore, CGI programs will call some Linux system commands to invoke R or Netpbm.

4.3 The User Interface

At present R is provided with a command line interface (CLI), which requires users to have a good knowledge of the R language. For proficient users, it is flexible and powerful.

However, novice users always find it difficult to work with a CLI, as they do not know exactly what can be done and how to do it. A GUI is typically a short cut for them and makes the learning curve shorter.

Many proprietary software packages have good GUIs. Minitab is one of them. It has very clear and concise interfaces for time series analysis methods. Some features of Minitab's GUI were mimicked in RForecast.

When creating a GUI for R in RForecast, the author attempted to make the interface both easy-to-use and easy-to-implement.

Each call to forecasting tools in RForecast will generate two HTML pages in turn. First, the user views the form in which several fields must be entered. This page can be called the input page. After the user clicks the *GO* button in the input page, all the parameters in it are submitted to CGI programs on the server. Then CGI programs will process these parameters and generate corresponding text or graphical results. Finally CGI programs will present these results in another page, the output page.

Since scripts that generate the two HTML pages are embedded into one CGI program, the CGI program needs to decide when it should process the input page or the output page. The hidden fields provided by JavaScript were used here to complete this task (see Appendices B.4 line 9-17).

Specifically speaking, a hidden field is inserted in the input page (see Appendices B.4 line 246). Since it is a hidden field, it can have a value of *Null* but the user cannot see the field and its value. CGI programs first read all parameters in the input page. If the value of the hidden field is *Null*, then the input page will be presented to the user. When the user inputs all the fields in the page, all the parameters will be submitted to CGI programs and the hidden field will then be set to *OK*. When CGI programs detect that the value of the hidden field is not *Null*, they will process the parameters and generate the output page. The user will see text or graphical results on the output page.

4.3.1 Main Features of the GUI

It is assumed that users have some knowledge of forecasting methods. But they need not know about the R language. In the GUI, the user is required to enter some parameters by typing a number, checking a checkbox or browsing a list of files. So the required operations

for users are very simple. Figure 4.4 shows the web page for Holt-Winters' smoothing method. We will take this page as an example to introduce the main features of the GUI as follows.

1. Organize groups of fields by sequence of use

To conduct Holt-Winters' smoothing analysis, the user is required to load data as a time series, set some parameters for the model, and choose favorite outputs. The user has to fill in several fields on the form. To make it easier for the user, we organize such fields as groups by sequence of use.

On the web page (see Figure 4.4), there are four groups that can be described as Input, Model fitting, Forecasting and Output, respectively. Each group has its task, for the convenience of users. When users fill in the form, they need to fill only in these fields that are related to their needs. For example, if the user does not want to make forecasts, he/she can just leave the Forecasting group blank, i.e., leave the *Generate forecasts* checkbox and leave the *Periods ahead of forecasting* textbox blank. They need not worry about this group at all.

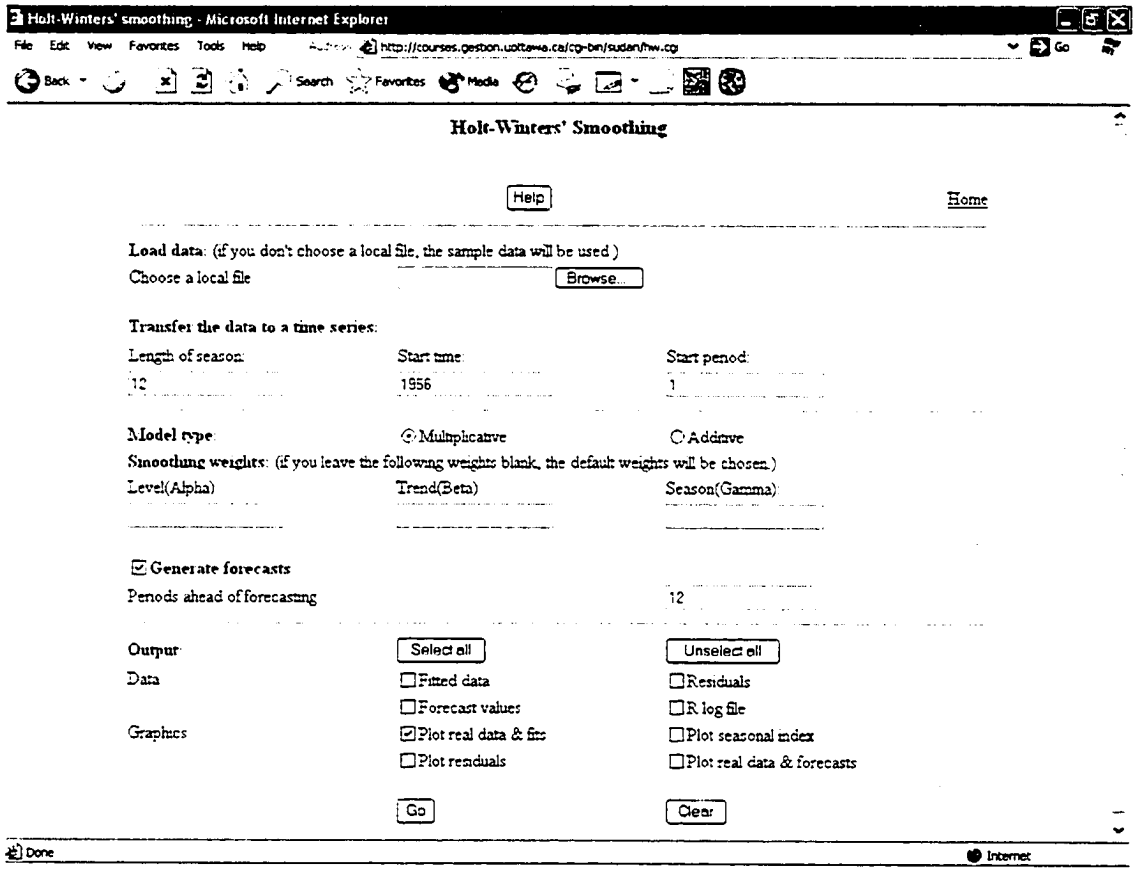

Figure 4.4. The web page for Holt-Winters' smoothing method

2. Make protected fields inaccessible

When filling in the form, novice users often do not know what fields are necessary to fill in and what fields are optional. They may waste time filling in non-necessary fields or choose inappropriate options. To avoid this kind of problem, we make some fields inaccessible to users.

In the above web page, if the user does not want to make any forecasts, he/she should not type in the *Periods ahead of forecasting* textbox or check the checkboxes of *Forecast values* and *Plot real data & forecasts*. Thus we protect these fields by disabling them whenever the *Generate forecasts* checkbox is unchecked.

Implement such functionality using JavaScript is not difficult. We control the protection by setting the *disabled* property to *True* or *False* and modifying the color style of

captions. When the *Generate forecasts* checkbox is unchecked, the *Periods ahead of forecasting* textbox and the checkboxes of *Forecast values* and *Plot real data & forecasts* are disabled, and the color of the corresponding captions becomes gray. Hence, the user cannot type or check these fields (see Appendices B.4 line 52-68).

Unfortunately, the *disabled* property is only defined by IE. Some other browsers such as NS and Mozilla do not support it. To open the web page using NS or Mozilla will return errors. This is so because the companies that produce different browsers provide their advanced features as fast as possible in order to obtain a loyal customer base (McFarlane et al., 1999).

To handle the compatibility problem, we have to detect the browser type. If the browser is IE 5.0+, the protection will be employed; otherwise, the protection will be hidden (see Appendices B.4 line 34-47).

3. Provide for defaults whenever necessary

In some cases, users may not know what input is acceptable or how to set appropriate parameters. For example, they cannot offer suitable time series. Or they do not exactly know what smoothing parameters are appropriate. How can they still make forecasting analyses?

Here the interface provides for defaults whenever possible. It offers a set of data as sample data in order for users to learn forecasting methods even without time series in hand. The sample data (Hyndman, undated) is the monthly production (megalitres) of beer in Australia from January 1956 to August 1995. The total number of observations is 472. It is a typical seasonal time series.

Similarly, it sets default values for smoothing parameters such as α, β and γ, if the users leave the corresponding fields blank. The R function *HoltWinters( )* has arguments that can be used to set defaults for smoothing parameters. If α, and/or β and/or γ are set to *NULL*, the function will try to find the optimal values for them by minimizing the squared one-step prediction error. People can make use of the optimization provided by R to set defaults. Thus novice users need not worry about how to set parameters.

It would be helpful to users if the defaults used by R were displayed in our screen (Figure 4.4). However, these are not readily available to the user interface code until R is invoked, which at the stage of user parameter entry is still pending.

4. Provide for simple prompts and instructions

In order to facilitate the model fitting, the interface provides for concise prompts and instructions when possible. In the same web page, users may not exactly know what *length of season*, *start time* or *start period* is. It puts some example values in the textboxes when the user loads the web page. In this way, the user can get concepts about these terms and then figure out how to set his/her own parameters.

Similarly, it adds some explanation to captions in order to instruct users how to fill in the fields. For example, it adds *if you don't choose a local file, the sample data will be used* to the caption *load data*. If users have no appropriate data in hand, they would be happy to know that the sample data would be loaded and used to fit the model.

The ways to offer the supporting information will be discussed in Section 4.5.

4.3.2 Data Validation on the Client Side

Though CGI is applied to process users' input, there is a fast and easy way to do data validation. JavaScript makes data validation available and easy-to-implement on the client side rather than on the server side. In this case, the parameters entered by the user can be checked for whether they are appropriate for model fitting. If mistakes exist, such as typing mistakes, decision errors or omission of required fields, the parameters will not be sent to the remote server. "Because validation takes place on the client machine, there's no delay for contacting a remote server. The user gets quicker responses, and network bandwidth and server processing power are both conserved." (Netscape Communications Corporation, 1997)

This is beneficial not only to users but also to developers. My experience is that it seems relatively easier and simpler to code and debug with JavaScript than with CGI scripts such as Perl. In addition, JavaScript supports Dynamic HTML (DHTML), which refers to the use of the Document Object Model (DOM), Cascading Style Sheets (CSS), and client-

side scripting to make Web pages more interactive (Feldman, undated). Thus the interface created by JavaScript can be very user-friendly.

DOM, "a programming interface specification being developed by the World Wide Web Consortium (W3C), lets a programmer create and modify HTML pages and XML documents as full-fledged program objects. " (http://whatis.techtarget.com)

CSS is "a data format used to separate style from structure on a Web page." (http:// www.marketingterms.com/dictionary/)

4.4 Main Modules

The interface includes five main modules as follows:

- Input module;
- Holt-Winters' smoothing module;
- Time Series Decomposition module;
- ARIMA modeling module;
- Output module.

First, users need to choose a data file from their local computers and enter relevant parameters. Then one forecasting tool needs to be chosen from such three forecasting tools as Holt-Winters' smoothing, Time Series Decomposition or ARIMA modeling. Finally corresponding data results and graphics are outputted. The relationship among the main modules is shown in Figure 4.5.

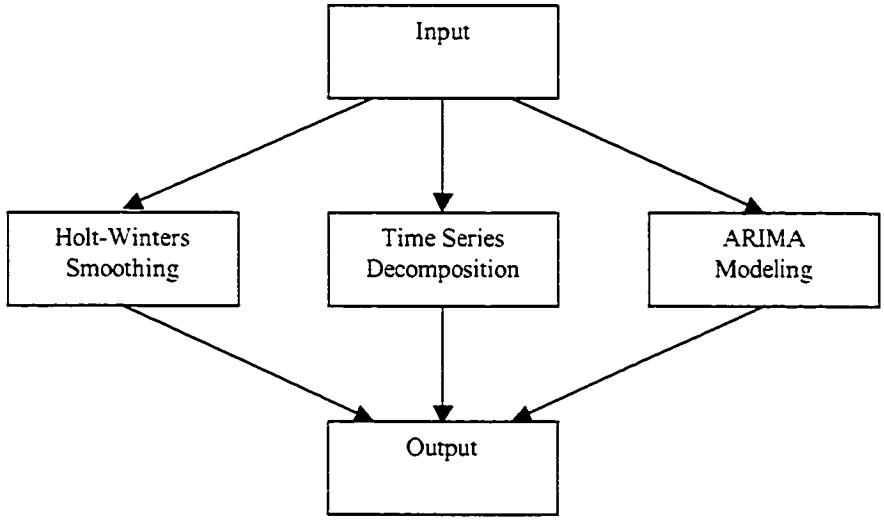

Figure 4.5. The relationship among main modules

4.4.1 Input Module

This module is designed for entering a data file and parameters from the users. First, the user is required to choose a data file from the local computer and enter some parameters on the form. If the user does not choose a local data file, the sample data will be used. After the user clicks the *Go* button, these inputs will be validated to make sure that the data in the client-side file (if applicable) can be loaded to R and converted to a time series object. If the validation fails, an error message will be sent to the user in HTML format and the user can try again. If the validation succeeds, the data in the client-side file will be copied to a text file on the server. Then R is invoked in batch mode, and the data saved in server-side file is loaded to R and converted to a time series object.

Figure 4.6 shows the flow chart of the module.

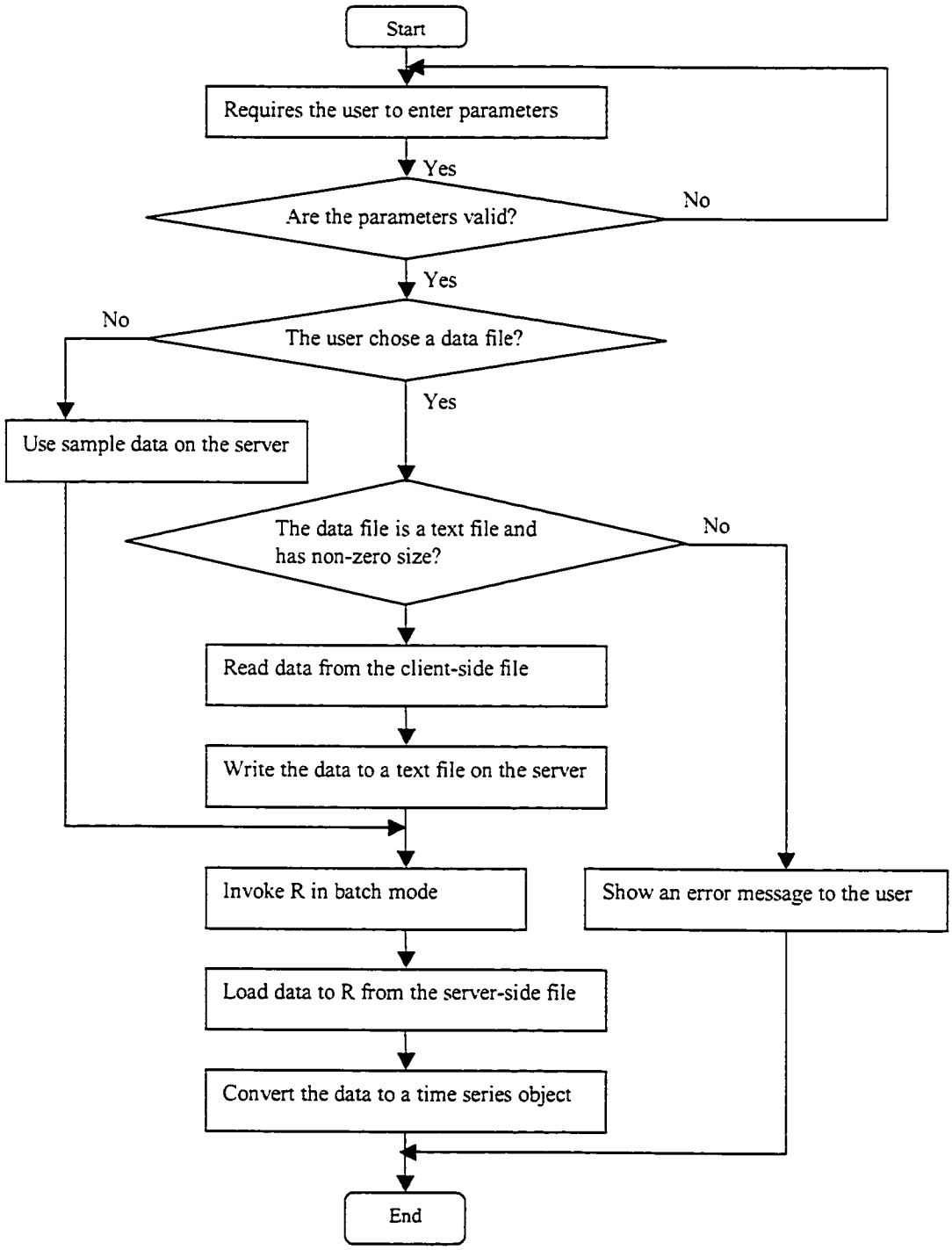

Figure 4.6. The flow chart of Input module

Here, the parameter validation is conducted on the client side. If the user chooses a local file, then the file name will be sent to the server and CGI programs will validate the file type. The second validation is conducted on the server side.

R provides facilities to read data into the system. It can read data from many kinds of files. The most common file type is the text type. With the aid of R add-in packages, R can also read data from special files such as S-Plus, SAS, Minitab, SPSS, Octave and even XML files. Users and developers should be aware that "statistical systems like R are not particularly well suited to manipulations of large-scale data." (R Development Core Team, 2002b) To make the RForecast implementation simple, data can only be read from text files at the moment.

If the data file is a text file, then CGI programs copy the content of the file to a temporary text file on the server side. This makes the text file easy for R to read in. Most statistical data files are in a spreadsheet-like format, that is, a rectangular grid with rows and columns. Thus the facility for reading data from a spreadsheet-like text files was adopted. The function *read.table( )* is the principal means of reading tabular data into R. The example usage syntax is as follows:

mydata<-read.table("c:/temp/beer.dat")

Here we must pay attention to two issues. One is that the *read.table( )* function is not suitable to read in very large numerical matrices, especially those with many columns. "It is designed to read data frames which may have columns of very different classes." (R Development Core Team, 2002b) The other issue is to make sure to use correct slash symbols in file paths. Some programs run on Windows systems may use the back slash "\". However, R, whether on Windows, Unix or Linux systems, uses the forward slash symbol "/".

Since forecasting analysis functionalities are supported by the *ts* package of R. We must remember to load the *ts* package whenever write R command files. This will avoid a common error.

Once we read data from a text file, the data needs to be transferred to a time series object. The *ts* package of R defines a class for time series objects. The function *ts( )* converts a numeric vector or matrix of the observed time-series values to time series objects. See the following example usage.

ts.mydata<-ts(mydata[,1],start=c(1956,1),frequency=12)

The argument *start* sets the start period at January 1956. The argument *frequency* sets the length of the seasonal period as 12.

To invoke R in batch mode, we use system commands as follows.

system("Rpath/R --vanilla <commands.R >log.txt")

Here, *Rpath* refers to the path name of R. *commands.R* refers to the R command file. *log.txt* refers to the R log file (see Appendices B.4 line 428).

4.4.2 Holt-Winters' Smoothing Module

This module is designed for fitting data or making forecasts using the Holt-Winters' smoothing method. First, the user is required to enter some parameters on the form. After the user clicks the *Go* button, these inputs will be validated to make sure that parameters are accepted by the Holt-Winters' tool in R. If the validation fails, an error message will be sent to the user in HTML format and the user can try again. If the validation succeeds, R is invoked in batch mode. The data is fitted using the Holt-Winters' method. According to whether the user wants to make forecasts or wants to save data results or graphics, corresponding outputs are generated.

Figure 4.7 shows the flow chart of the module.

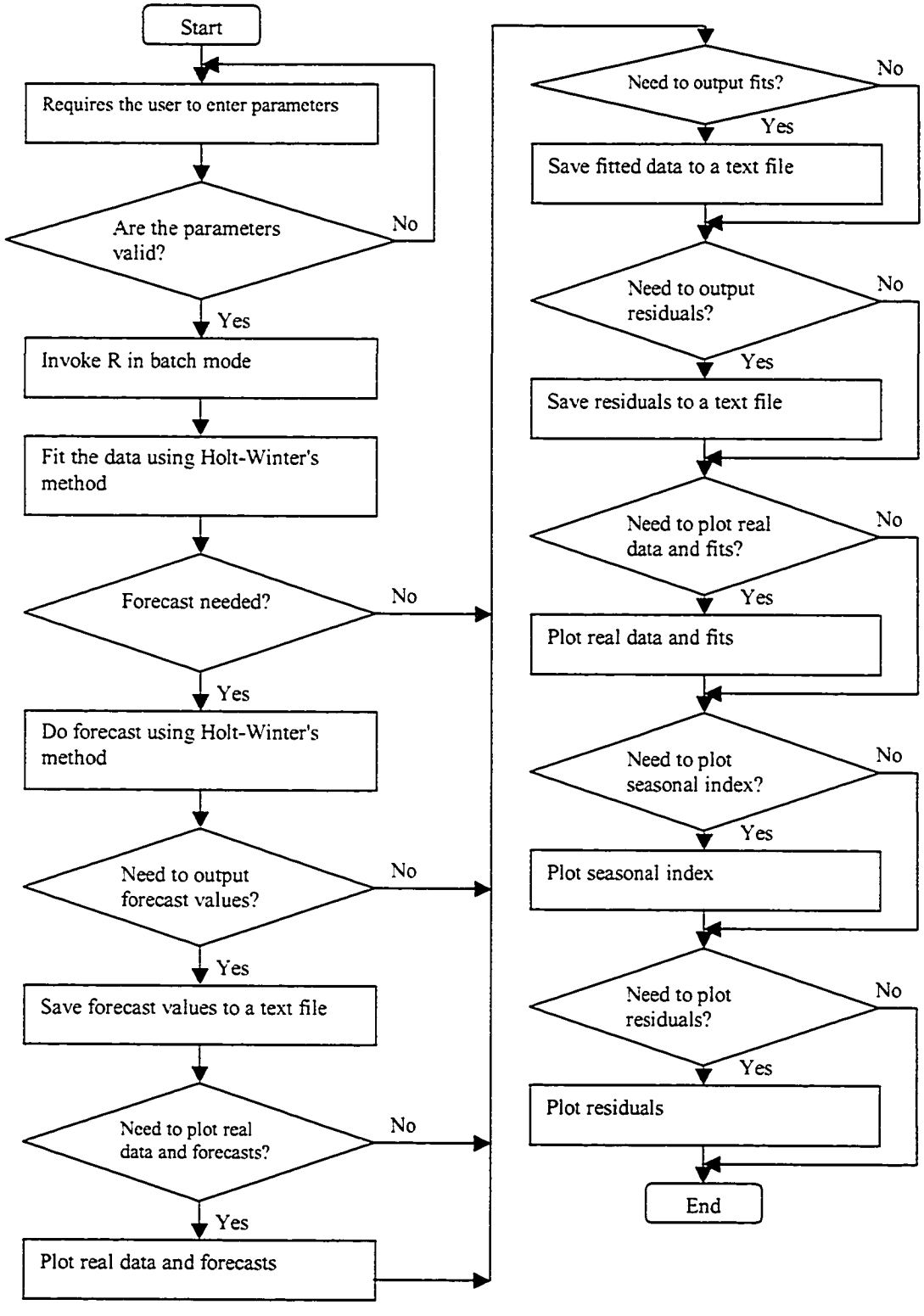

Figure 4.7. The flow chart of Holt-Winters' smoothing module

The *ts* package of R provides a function, *HoltWinters( )*, to compute Holt-Winters Filtering of a given time series. The example usage reads as follows:

hw.mydata<-

HoltWinters(ts.mydata,alpha=0.2,beta=0.3,gamma=0.1,seasonal=multiplicative)

Here, the argument *seasonal* is used to select an additive or multiplicative model. Note that it should be set differently according to the version of R. For the Windows version of R, the argument can only be *additive* or *multiplicative*, while for R in the Linux version, only *Additive* or *Multiplicative* can be recognized. That is, capitalization is important.

The argument *beta* is the smoothing constant for the trend smoother while *gamma* is for the seasonal smoother. As the author mentioned in Section 2.3.1, R uses this notation, which may be different from that in some books.

Depending on different model types, the filtering uses different formulas as follows.

Additive model:

$$S(t)=\alpha*(Y(t)-I(t-L))+(1-\alpha)*(S(t-1)+b(t-1))$$
$$b(t)=\beta*(S(t)-S(t-1))+(1-\beta)*b(t-1)$$
$$I(t)=\gamma*(Y(t)-S(t))+(1-\gamma)*I(t-L)$$

Multiplicative model:

$$S(t)=\alpha*Y(t)/I(t-L)+(1-\alpha)*(S(t-1)+b(t-1))$$
$$b(t)=\beta*(S(t)-S(t-1))+(1-\beta)*b(t-1)$$
$$I(t)=\gamma*Y(t)/S(t)+(1-\gamma)*I(t-L)$$

where: S(t) is the level smoother;

b(t) is the trend smoother;

I(t) is the seasonal smoother;

L is the season length.

The function tries to find the optimal values of *alpha*, *beta* or *gamma* by minimizing the squared one-step prediction error if values are not specified (R Development Core Team, 2002c).

The function creates an object of class *HoltWinters*, a list with components such as the fitted time series, the seasonal indices, and the final sum of squared errors (SSE) achieved in

optimizing. Hence, it is easy to output the fitted data, the seasonal indices and SSE. The function *residuals.HoltWinters( )* generates the residuals for Holt-Winters filtering.

The *ts* package provides a prediction function for Holt-Winters method, function *predict.HoltWinters( )*. The example usage reads as follows:

fhw.mydata<-predict(hw.mydata, n.ahead=12)

Depending on different model types, the prediction function uses different forecasting formulas. The k-period ahead forecast takes the following forms.

Additive model: $F(t+k) = S(t) + k*b(t) + I(t+1+(k-1) \bmod L)$

Multiplicative model: $F(t+k) = (S(t) + k*b(t))*I(t+1+(k-1) \bmod L)$

where L is the season length.

If the user wants to output results data, such as fitted data, residuals or forecast values, R has a facility for exporting data. First, the results data should be converted to a matrix and then a vector. Next the function *write.table( )* can write a matrix or data frame to a text file with row and column names. Thus R can add appropriate column names to a text file. This is a good way to remind the user what results are included in the file. The following is an example usage.

ma.fore<-data.matrix(fore. mydata)

ve.fore<-as.vector(ma.fore)

write.table(ve.fore, file="foredata.txt", quote=FALSE, row.names=FALSE, col.names=c("forecasts"))

Note that the *write.table( )* function is not suitable for writing out very large matrices.

Graphical outputs are also provided to users. "Graphical facilities are an important and extremely versatile component of the R environment. It is possible to use the facilities to display a wide variety of statistical graphs." (R Development Core Team, 2002a) These facilities can be used in both interactive and batch modes, so there are two approaches to generating graphical outputs.

One is via the interactive model. When R is invoked in batch mode with a R command file, the echo session can be saved to a Postscript file. In other words, the Postscript file includes the echoed commands and numerical or graphical results. Thus graphical outputs can be abstracted from the Postscript file. This approach was adopted in the Rweb package that was created by Jeff Banfield, an Associate Professor of Statistics at Montana State

University. Rweb was a simple text entry form that returns output and graphs (see http://www.math.montana.edu/Rweb/Rweb.general.html).

The other approach is to output graphics directly to a file. There are many graphics devicers in R, such as Postscript, BMP, JPEG. Since R supports the Postscript driver by default on both the Windows and Linux platform, it is convenient to use the Postscript driver to store graphics in Postscript files.

Both the above approaches work well. In our case, the second one was chosen in order to show users how to output graphics directly to a file (see Appendices B.4 line 369-377). The example is given as follows:

```
#plotting the real data and forecasting values
postscript(file="realfore.ps", width=30, height=30)
ts.plot(ts.mydata, fore.mydata, col=c("blue","red"))
dev.off()
```

4.4.3 Time Series Decomposition Module

This module is designed for fitting data or making forecasts using the Time Series Decomposition method. First, the user is required to enter some parameters on the form. After the user clicks the *Go* button, these inputs will be validated. If the validation fails, an error message will be sent to the user in HTML format and the user can try it again. If the validation succeeds, R is invoked in batch mode. The data is fitted using the Time Series Decomposition method. According to whether the user wants to make a forecast or wants to save data results or graphics, corresponding outputs are generated. Finally, the plot of real data and fitted data is shown to the user.

Figure 4.8 shows the flow chart of the module.

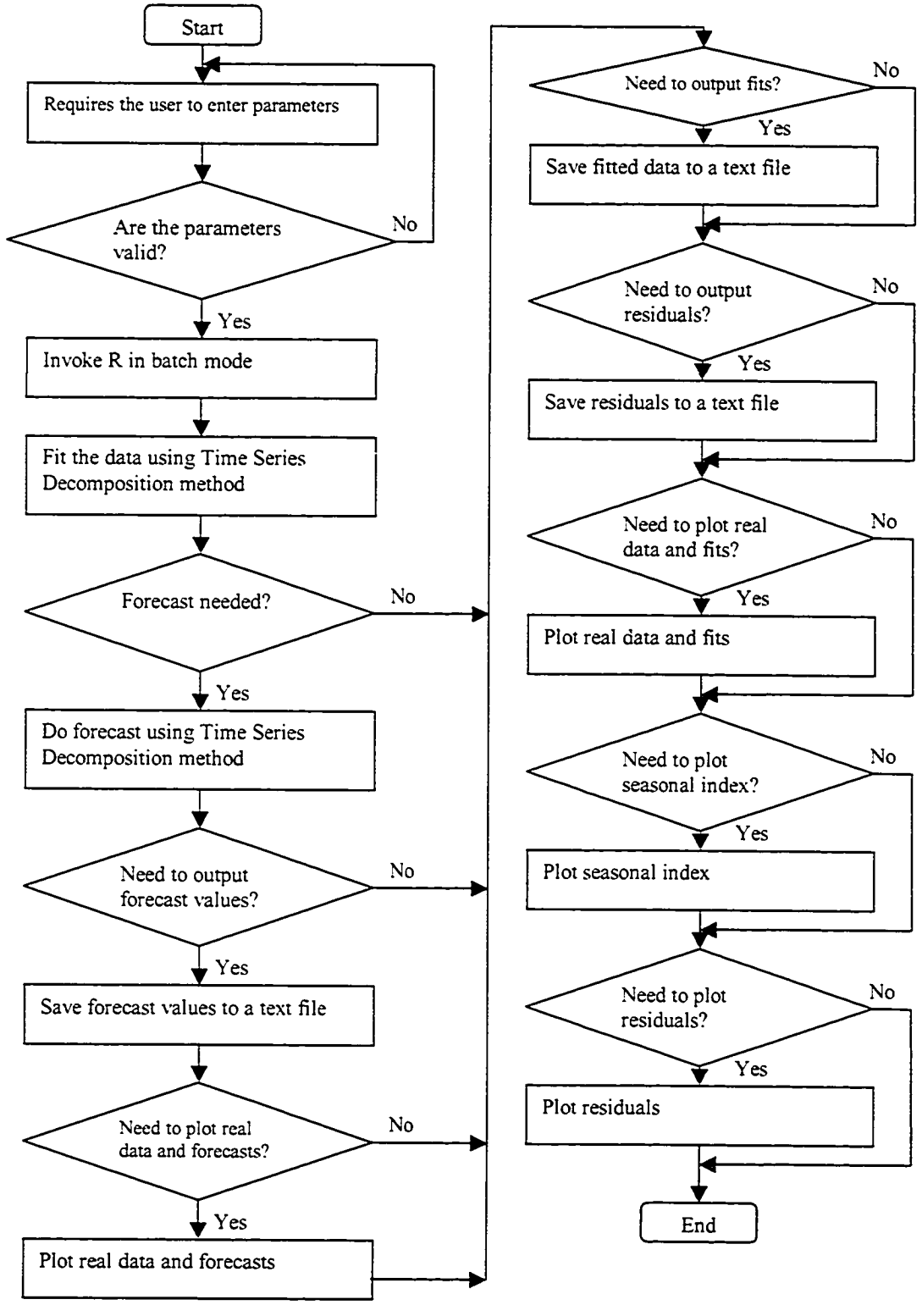

Figure 4.8. The flow chart of Time Series Decomposition module

The *ts* package of R provides a function to compute a classical Time Series Decomposition (TSD). The function *decompose( )* decomposes a time series into seasonal, trend and irregular components using moving averages. It can deal with additive or multiplicative seasonal component as follows.

Additive model: $Y(t) = T(t) + S(t) + e(t)$

Multiplicative model: $Y(t) = T(t) * S(t) + e(t)$

Where: T(t) is the trend component;

S(t) is the seasonal component;

e(t) is the irregular component.

The resulting value is an object of class *decomposed.ts* with components such as the seasonal and trend components and seasonal indices. Thus seasonal indices are easy to output.

Unfortunately, the *ts* package does not provide a prediction function for TSD. Since TSD is a very common tool for forecasting analysis, the lack is viewed as inconvenient for forecasters. So the RForecast interface attempted to provide the prediction facility for TSD using the following formula.

Additive model: $F(t+k) = (a + b*(t+k)) + s(t+k)$

Multiplicative model: $F(t+k) = (a + b*(t+k)) * s(t+k)$

where: a is the intercept of the trend equation;

b is the slope of the trend equation;

s is the seasonal adjustment.

To create such a prediction function for TSD, two issues should be considered: the trend equation and seasonal adjustment.

R has a function, *lm( )*, which is used to fit linear models. So it is easy to formulate a trend equation for a time series (see Appendices B.5 line 300-302). But to generate a seasonal adjustment for a given period is not so simple.

First, seasonal indices can be obtained from the result of function *decompose( )*. Then we can formulate the seasonal adjustment series according to the season length. Finally, we generate the seasonal adjustment for the specified period (see Appendices B.5 line 305-315).

Fitted data and residuals have to be computed according to both the trend equation and the seasonal adjustment.

Note that forecasts and fitted data should be transferred to objects of time-series type, before they can be plotted using the plotting function *ts.plot( )* (see Appendices B.5 line 354-361). This function is used to plot multiple time series in one graph.

4.4.4 ARIMA Modeling Module

This module is designed for fitting data or making forecasts using the ARIMA modeling method. First, the user is required to enter some parameters on the form. After the user clicks the *Go* button, these inputs will be validated. If the validation fails, an error message will be sent to the user in HTML format and the user can try again. If the validation succeeds, R is invoked in batch mode. The data is fitted to the ARIMA model. According to whether the user wants to make forecasts or wants to save data results or graphics, corresponding outputs are then generated. Finally the plot of the real data and fitted data is shown to the user.

Figure 4.9 shows the flow chart of the module.

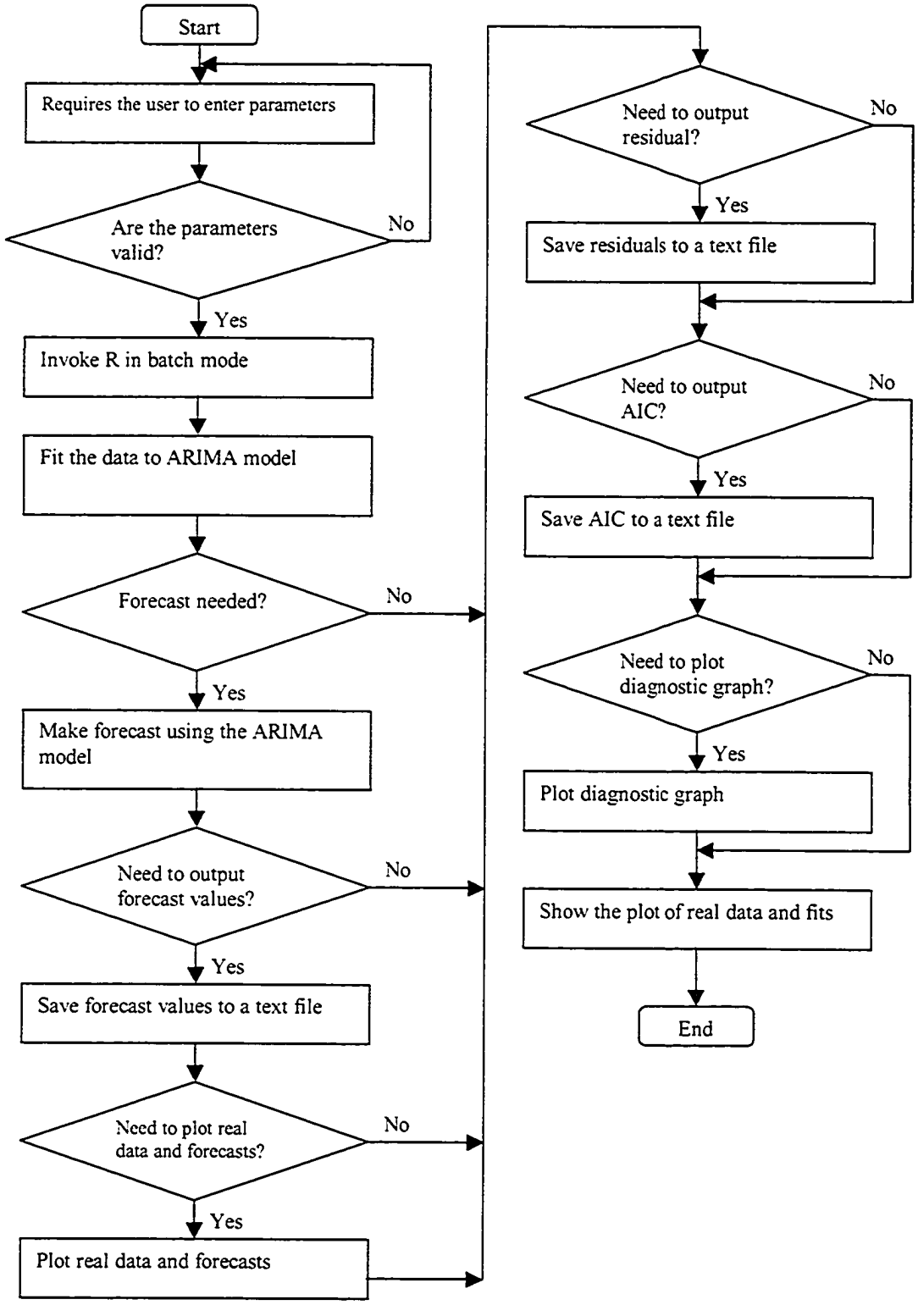

Figure 4.9. The flow chart of ARIMA modeling module

The *ts* package of R provides a function for ARIMA modeling of time series. The example usage for the function *arima( )* reads as follows:

arima.mydata<-arima(ts.mydata,order=c(1,0,1),seasonal=list(order=c(0,0,1)), include.mean=TRUE)

The argument *order* specifies the non-seasonal part of the ARIMA model: the three components are the AR order (p), the degree of differencing (d), and the MA order (q). The argument *seasonal* sets the seasonal part of the ARIMA model. The argument *include.mean* specifies whether the model includes a mean term.

Different software packages define different signs for the AR and/or MA coefficients. The definition in R is

$$Y(t) = AR(1)Y(t-1) + ... + AR(p)Y(t-p) + e(t) + MA(1)e(t-1) + ... + MA(q)e(t-q)$$

The function creates a list of class *Arima*, which includes a vector of AR, MA and regression coefficients and the Akaike's Information Criteria (AIC) value.

Forecast values can be completed using the function *predict.Arima( )*.

The *ts* package has a plotting function *tsdiag* for plotting time series diagnostics, which is helpful for diagnostic checking.

4.4.5 Output Module

The module is designed for outputting both data and graphical results. First, it checks whether there exists a graphical file in PS format. If so, convert PS format to PPM format and then convert PPM format to GIF format. The corresponding PS and PPM files will be deleted. After the PS graphics have been converted and deleted, these GIF graphics will be linked within the HTML page of the interface for the user. Then it checks whether there exists a text output file. If so, it retrieves data from the text file and sends it to the interface page. The text file will then be deleted.

Figure 4.10 shows the flow chart of the module.

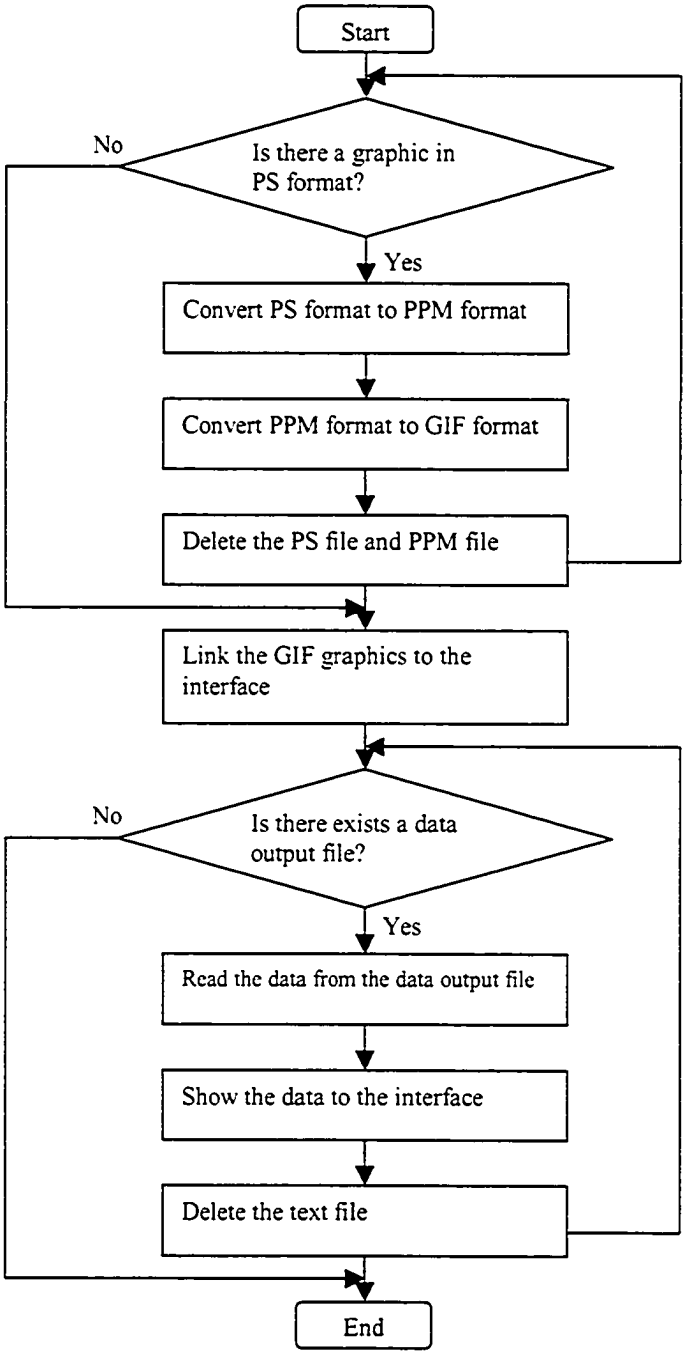

Figure 4.10. The flow chart of the Output module

Users may obtain two types of outputs. One is text files including fitted data, residuals, forecast values or R logs. The other is graphical files including plots. To make it

easy for users to view outputs and save them for future use, the outputs should be processed appropriately.

For text files, it is easy to show the content in HTML pages, while for graphical files, it is not so simple. Since most browsers do not support Postscript graphics displaying, Postscript format has to be converted to GIF format that can be displayed on most browsers.

Here, we refer to the conversion mechanism introduced by Jeff Banfield. In Rweb he implemented graphics conversion using the Netpbm toolkit. He explained the conversion in his paper "Rweb: Web-based Statistical Analysis" (Banfield, 1999).

"Netpbm is a toolkit for manipulation of graphic images, including conversion of images between a variety of different formats. There are over 220 separate tools in the package including converters for about 100 graphics formats." (see Netpbm Home page, http://netpbm.sourceforge.net/README) The improvement and modification of the package is supported by programmers all over the world. The programs work with a set of graphics formats such as PBM, PGM, PPM and PAM. The first three formats are also called PNM. The program *pstopnm* can convert PostScript to PNM. The program *pbmtogif* can convert PBM to GIF. In RForecast, CGI programs call the two programs to convert Postscript to GIF. Figure 4.11-14 are example graphics processed using Netpbm. The color overlap cannot be printed in black and white.

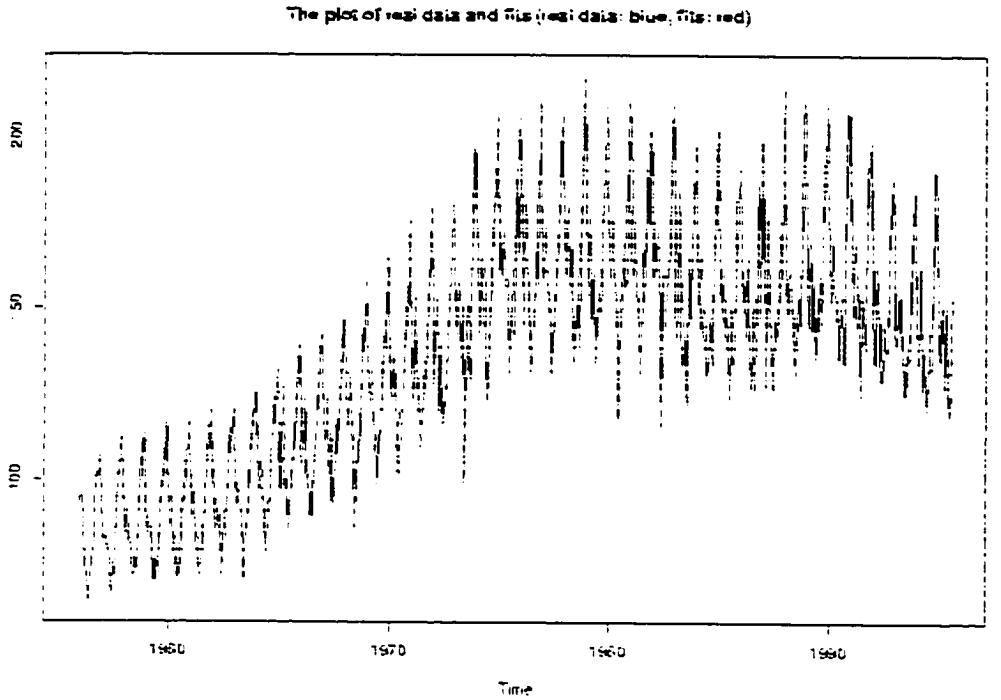

Figure 4.11. The plot of the real data and fits via the Holt-Winter's method (The color overlap cannot be printed in black and white)

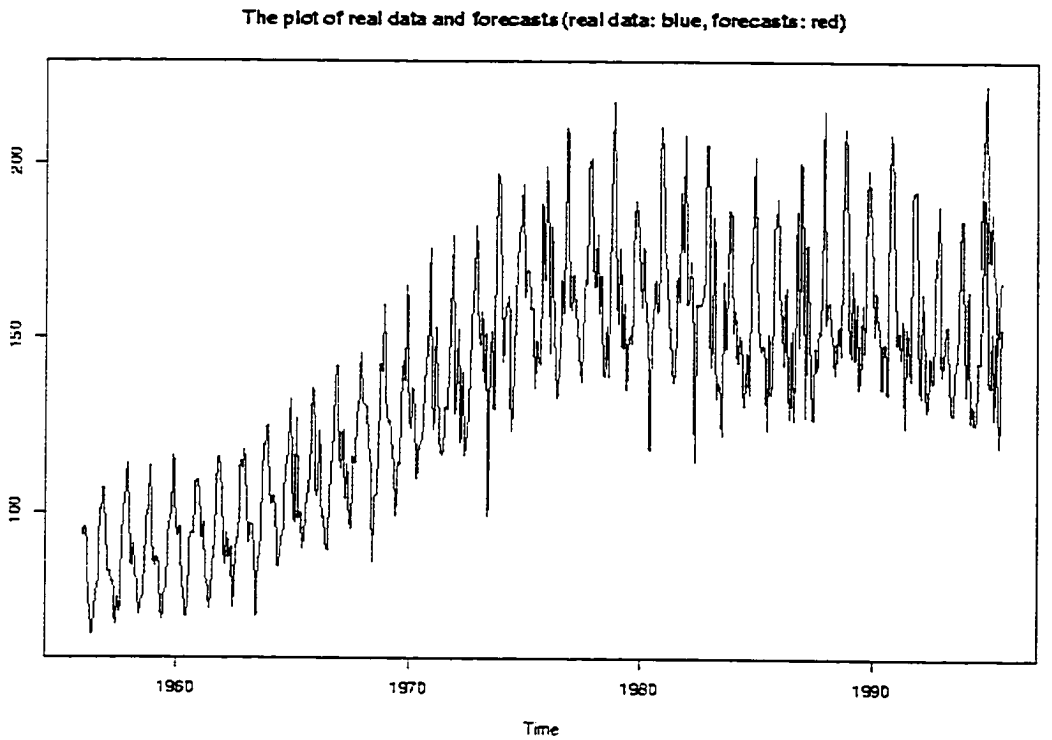

Figure 4.12. The plot of the real data and forecasts via the TSD method (The color overlap cannot be printed in black and white)

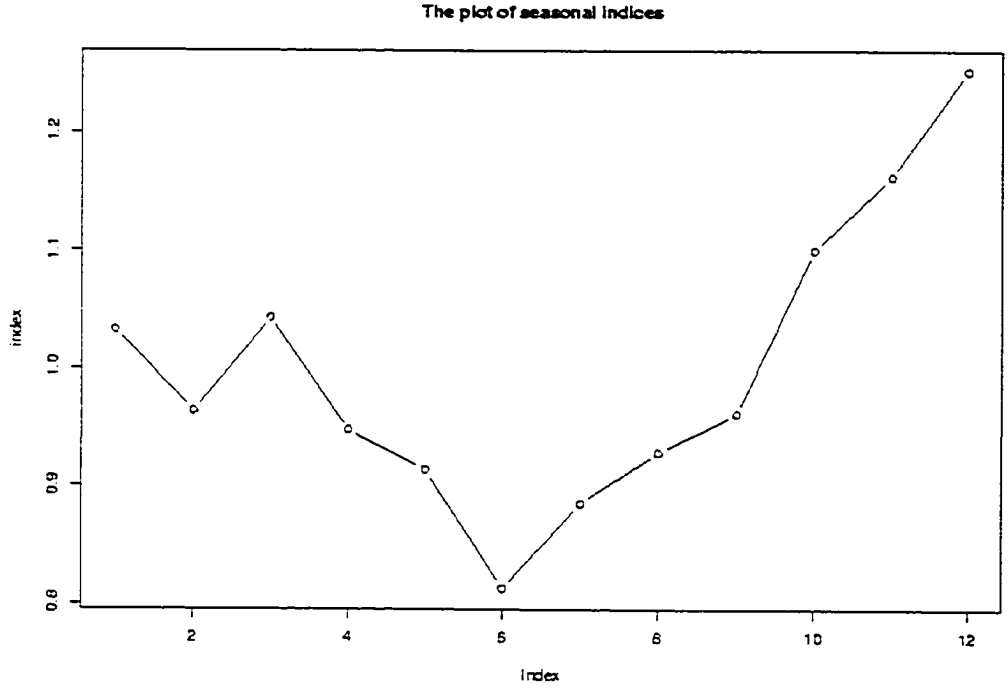

Figure 4.13. The plot of the seasonal indices via the TSD method

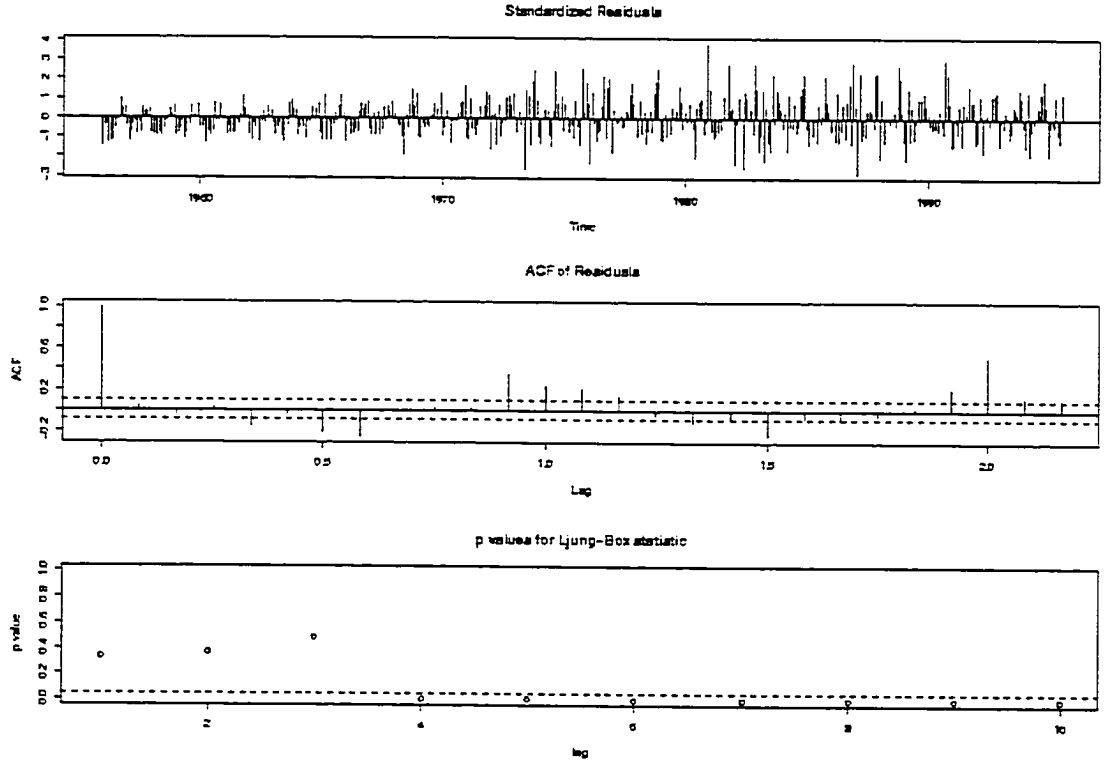

Figure 4.14. The diagnostic graph for the ARIMA modeling method

There has been a controversy on GIF format for more than seven years. Michael C. Battilana introduced the controversy in his article.

> "Between 1987 and 1994, GIF (Graphics Interchange Format) peacefully became the most popular file format for archiving and exchanging computer images. At the end of December 1994, CompuServe Inc. and Unisys Corporation announced to the public that developers would have to pay a license fee in order to continue to use technology patented by Unisys in certain categories of software supporting the GIF format. These first statements caused immediate reactions and some confusion. As a longer term consequence, it appears likely that GIF will be replaced and extended by new file formats, but not so before the expiration of the patent which caused so much debate." (Battilana, 2002)

PNG (Portable Network Graphics) is one of patent free formats designed to replace to GIF format. Today many web browsers including IE, NS and Mozilla have supported PNG images. And many people advocate to use PNG or JPEG (Joint Photographic Experts Group) formats rather than GIF format. The NETpbm toolkit has programs such *pnmtopng* or *pnmtojpeg* that can convert PNM to PNG or JPEG. Thus it is also feasible to convert PS to PNG or JPEG in the RForecast implemenation.

4.5 Supporting Information

Most users need supporting information when they work with software, free or otherwise. They may be unclear what results they can expect, or they may not know how to choose a parameter. Sometimes a procedure could be interrupted or halted. They want to know what caused the failure. Novice users or students especially need such helpful messages.

Due to the main features such as convenience, quickness and ease of use, many users prefer on-line help rather than traditional hard-copy documentation. On-line help can provide users with real-time supporting information. It offers users clear and simple identification and instructions. It also gives users good feedback on certain procedures after errors. Deborah Mayhew described the following features of on-line help in her book:

"it can't be lost or damaged.

it does not require disk space and place holding.

it is faster and cheaper to update than hard-copy documentation." (Mayhew, 1992)

"Modern graphical user interfaces provide a variety of mechanisms for the user to gain access to on-line help and for the system to present help." (Constantine and Lockwood, 1999) Considering the issue of both ease of use and ease of implementation, the following ways to present supporting information were attempted in the RForecast interface.

- Descriptions at the cursor;
- Descriptions on the status bar;
- Hints inline;
- Instructions in a separate window;
- Error messages as feedback.

1. Descriptions at the cursor

Generally, the current position of the cursor attracts the user's attention most easily. It is a good place to present supporting information, especially identification messages. Hence, some identification descriptions are put near the position of the cursor. The following screen image (Figure 4.15) is an example. Users may get confused when they fill in the parameter *trend*. If the mouse is over the textbox, a description about how to set it will be shown to the left of the cursor. The color of the description text is set to red in order to attract the focus of users' attention. It can help users to enter appropriate parameters.

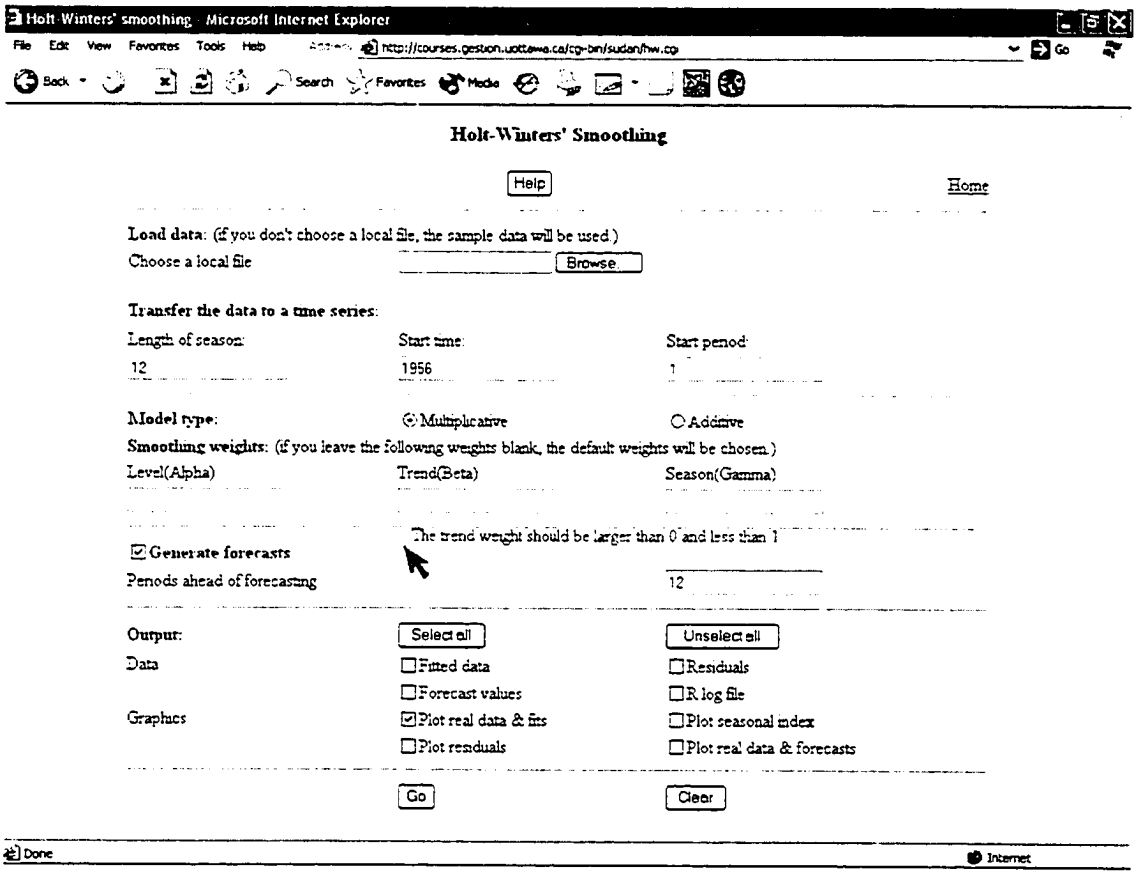

Figure 4.15. The description at the cursor for Holt-Winters' smoothing method

This implementation in IE is easy with the tag SPAN using JavaScript. Unfortunately, this feature cannot be supported by some browsers such as NS or Mozilla. The compatibility problem is a big problem in such implementations. The browser type has to be screened. Only if the browser is IE5.0+ can the above feature be implemented.

2. Descriptions on the status bar

Unlike messages at the cursor, status bar messages are a standard feature of Windows-based interfaces (Constantine and Lockwood, 1999). Since the position of the status bar is fixed at the bottom of the screen, it is a common way to offer identification or suggestion messages on the status bar. For example, when the user puts the mouse on the link with the caption of Holt-Winters' method, information on the status bar will be changed from *RForecast* to

Click here to conduct forecasting analysis using Holt-Winters' method as in the following illustration (Figure 4.16).

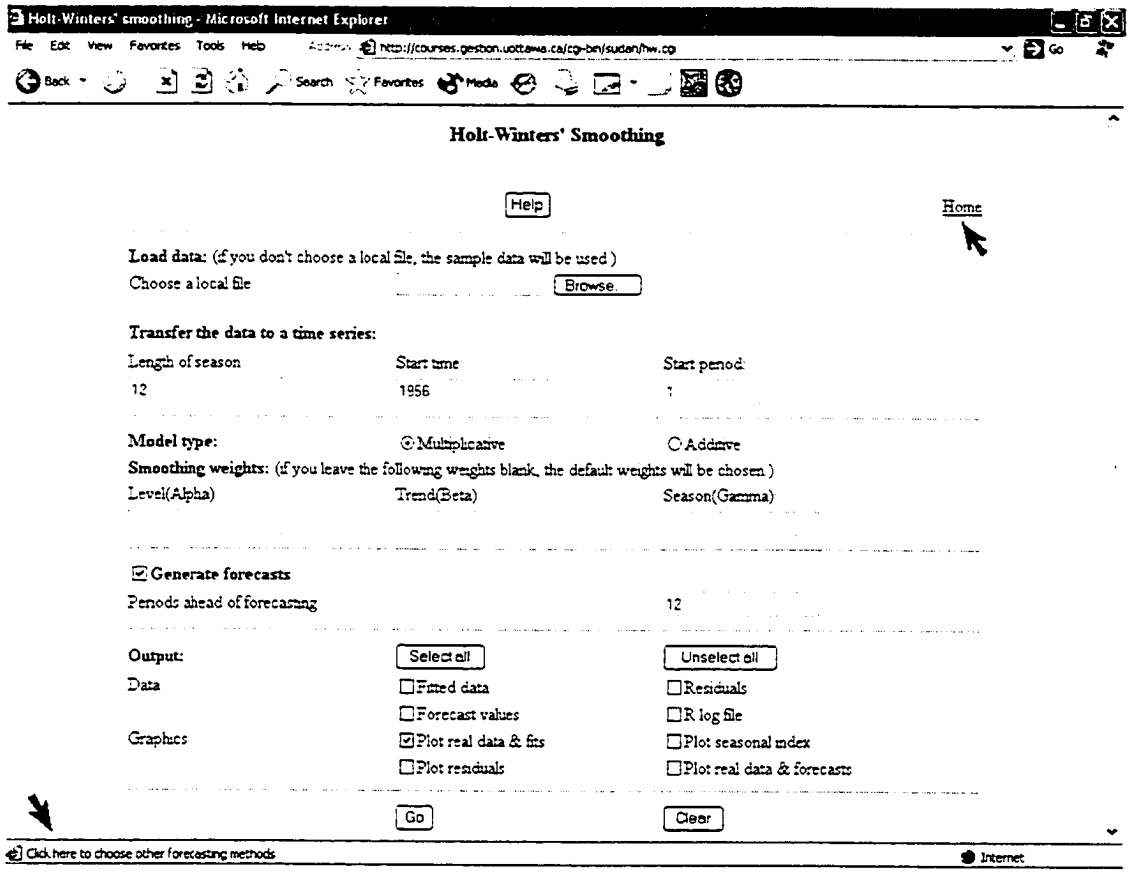

Figure 4.16. The description on the status bar for Holt-Winters' smoothing method

However, some users perhaps ignore messages on the status bar because of its low position on the screen. So important instructions should not be put on the status bar.

3. Hints inline

Brief and concise hints or prompts regarding valid inputs can provide users with good instructions to correctly fill in the form. This kind of support is very visual and clear. It is an effective method to offer help messages. In the interface, hints are provided when possible.

As mentioned in Section 5.1, example values are put into textboxes when the user loads certain web page. This helps the user decide how to set corresponding parameters.

Similarly, concise explanations are added to some captions to instruct users to fill in fields. For example, *if you leave the following weights blank, the default weights will be chosen* is added to the caption *smoothing weights* (Figure 4.17). If users have no idea about how to choose smoothing weights, the weights will be set to *NULL* by default, and the optimal values of α and/or β and/or γ will be found by minimizing the squared one-step prediction error.

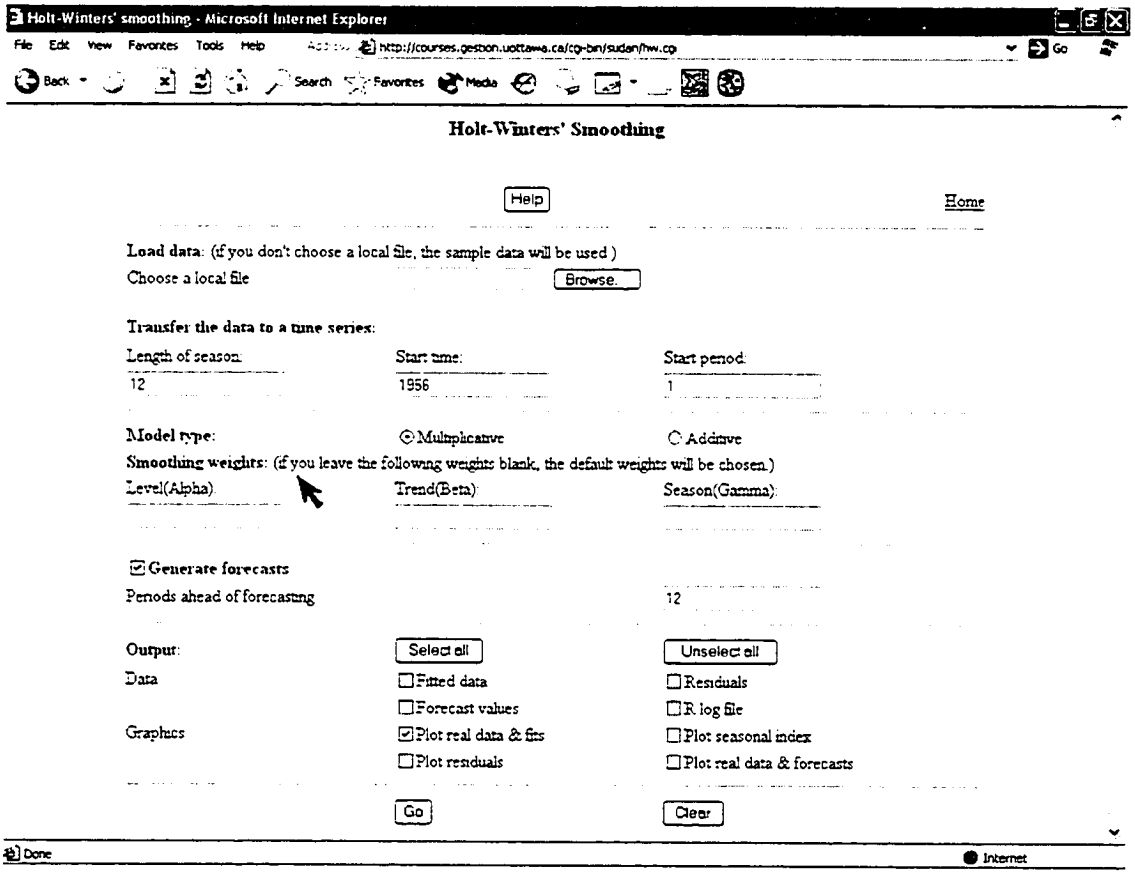

Figure 4.17. The hint for Holt-Winters' smoothing method

4. Instructions in a separate window

In some cases, users may want to have detailed knowledge of certain methods. These messages are displayed in a separate window, independent of the active window. This is a traditional way to present supporting information.

In this case, a *help* link is provided in every fill-in form. Basic concepts, theoretical background or step-by-step procedures are presented in a separate window. By reading the

messages, users can learn when and how to apply specific forecasting methods. This is a good way to offer users detailed supporting information.

5. Error messages as feedback

An error message is one of most important pieces of supporting information that users expect. Whenever a procedure is interrupted or halted, users need to find out the cause and make corrections if necessary.

If the user enters an unacceptable parameter, it is helpful to offer an error message upon completion of the field. This kind of support is provided, when possible, in RForecast. In the same web page discussed in Section 5.1, the values of smoothing weights are validated and error messages are presented if the values are not appropriate for model fitting. For example, if the user enters -2 for α, an alert window will be opened and an error message will be presented at the window (Figure 4.18). Meanwhile, after an input error is detected, the cursor will be relocated on the field that is in error. In the above example, since the value for α is unacceptable, the cursor will then return to the field. This will facilitate error correction. These implementations are easy using JavaScript.

Similarly, if the data file chosen by the user is not suitable to perform model fitting using a certain method, an error message will be presented as feedback (Figure 4.19). The error message may give some possible explanation of what can cause the error. Thus the user can conduct error analyses and figure out where the error is. This is beneficial for novice users.

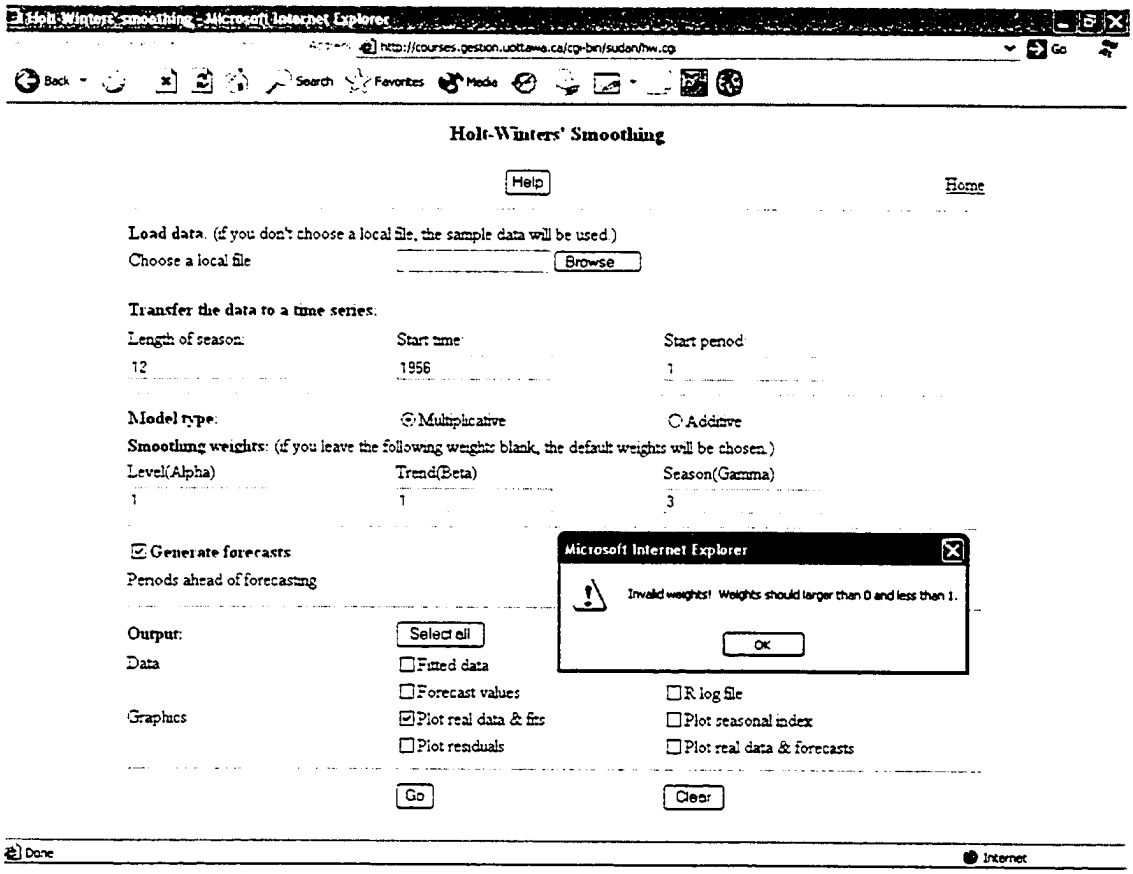

Figure 4.18. The alert window for Holt-Winters' smoothing method

6. The log file of the underlying computational system is available

We have chosen to allow the user to display the log file of the computational system (in this case R) for the calculations that are carried out. This provides a record of what has been done, and may include messages that describe details of the methods. We consider this to be supporting information, in that it is helpful in situations where the results are less than fully satisfactory and we must consider alternative forecasting tools. Because the log file is rather specific to the computational system and the method, we have chosen not to include an example here.

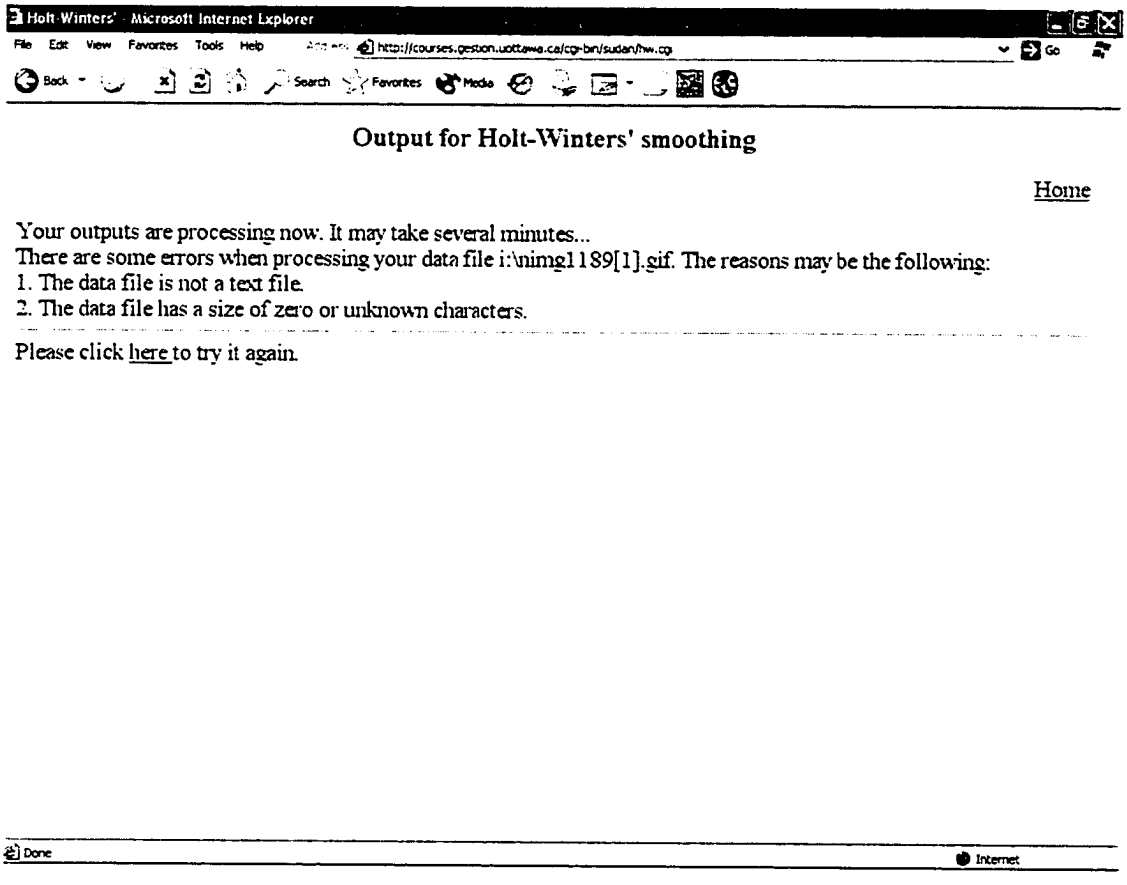

Figure 4.19. The error message for Holt-Winters' smoothing method

4.6 Summary

The implementation of RForecast demonstrates a feasible and effective approach to facilitating and simplifying the use of R. The approach can be generalized as follows:

- Creating a web-based GUI using Perl and JavaScript;
- Retrieving data inputs and model parameters from the client;
- Copying the content of client-side data files (if applicable) to text files on the server;
- Creating R command files that calls R functions;
- Invoking R in batch mode and obtaining text or graphical results;
- Processing graphical results;
- Presenting text or graphical results to the user;
- Providing supporting information to help the user.

From the perspective of users, the interface is easy-to-use and convenient. First, the GUI is user-friendly. Users need no knowledge of R. They can get helpful supporting information and conduct forecasting analyses easily. They can learn the R language from R log files if they want. Secondly, the GUI is web-based. Users can browse and perform forecasting analyses via web browsers. The minimum requirement is only a JavaScript-capable browser. Furthermore, the interface provides users with both text and graphical outputs. It makes use of exporting facilities in R and processes graphical outputs using the Netpbm toolkit. The two types of outputs are presented in the interface. Users can view and save outputs for future use.

From the perpective of developers, the interface is also relatively easy-to-implement. My experience is that Perl and JavaScript are relatively easy to learn and understand, compared to C or FORTRAN. Neither need be compiled before running. The JavaScript scripts and R commands are actually embedded into the Perl code.

R is statistical software. To confirm the feasibility of the approach, the author attempted to implement another example interface for Scilab, which is scientific software. The next section will introduce its design and implementation. We also consider other possible approaches.

Chapter 5 Other Possible Implementations

To confirm the feasibility and efficiency of the web-based approach, this chapter illustrates an implementation for Scilab, which is scientific software. It then introduces a local implementation for R on the Windows platform. It also talks about issues on running web-based GUI locally through localhost. Finally it discusses other possible approaches to answer the research question.

5.1 A Web-based Implementation for Scilab

RForecast is the implementation for R. Since R is a statistical software package, it has functions that provide for statistical functionalities. It may have advantages people can use conducting forecasting analyses. Unfortunately, other scientific software packages may lack these advantages. It may be more difficult to conduct forecasting analyses using these scientific packages.

Scilab is a scientific software package for numerical computations and has been viewed as one of the free replacements for Matlab. Many people, especially researchers and educators, are exploring its wide application in many fields. Based on the same approach that is adopted in RForecast, a similar interface for Scilab, SciForecast, was created to confirm the feasibility of the approach. Only ARIMA modeling was implemented in the interface as a test case.

Thus, the GUI of the SciForecast interface is similar to that of Forecast. On the GUI form, the user is required to enter parameters on the form. After the *Go* button is clicked, the parameters are validated. If the validation is successful, the CGI program will create a Scilab command file and invoke Scilab in batch mode. The data results and graphical results are then presented on the screen in order for users to browse and save them.

Figure 5.1 illustrates the data flow through the system. The circles with numbers inside show the time sequence of data flows. Note that Netpbm is not used in this case. Scilab can easily output graphics in GIF format on both the Windows and Linux platforms. The graphic formats need not be converted.

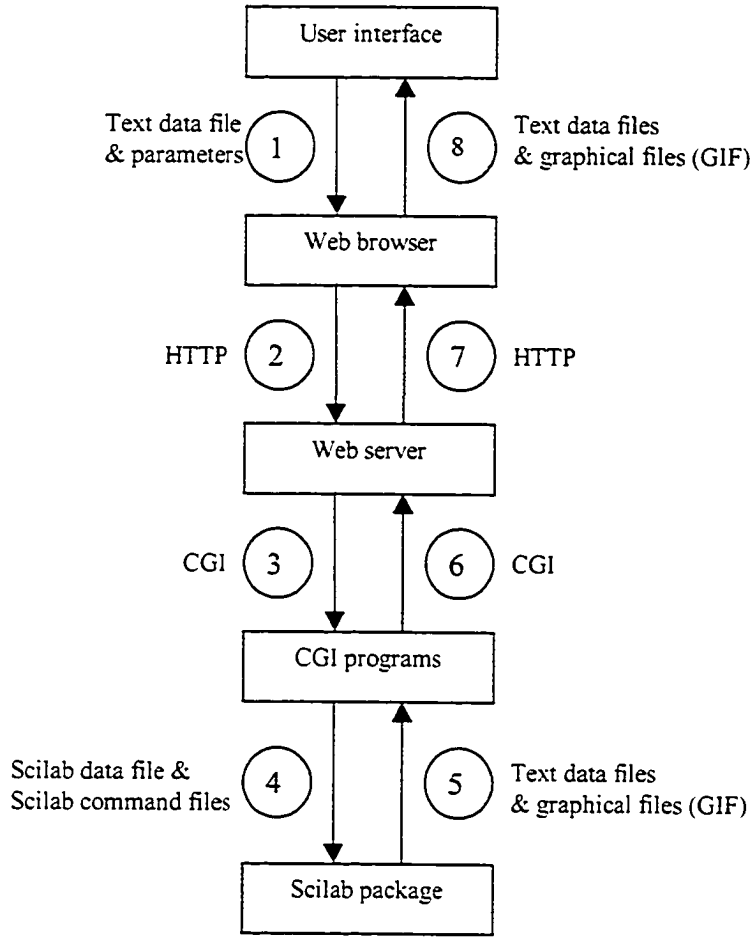

Figure 5.1. The data flows in SciForecast (the circles with numbers inside show the time sequence of data flows)

1. Loading data and exporting results

Similar to R, Scilab has convenient facilities for importing or exporting data. The function *read( )* is used to load data to Scilab from text files as in the following example of usage.

mydata=read("c:\temp\beer.txt",-1,1)

Note that *mydata* is a 1×n matrix. It may have to be transferred to a n×1 matrix by transposing in some cases.

The function *write( )* is used to export data from Scilab. The example usage reads as follows.

write("fitdata.txt", fitdata)

It is easy to export graphics from Scilab. The GIF driver can be used on either the Windows or Linux platforms. The procedure is similar to that in R. First, open the GIF driver; then specify the name of the graphical file. After plotting, turn off the driver. See the following example.

```
driver("GIF");
xinit("fitgraph.gif");
plot(fitdata);
xend();
```

2. Model identification

Unlike R, Scilab lacks built-in functions for ARIMA modeling. Though it has very powerful facilities for manipulating numerical matrices, Scilab is mainly focused on fields such as system control and signal processing applications. It has neither definition of time-series objects nor model fitting function related to time series models. This makes the implementation difficult.

Scilab has a toolbox for ARMA modeling and simulation. The toolbox is used for systems identification and simulation of ARMAX (Auto Regression Moving Average with eXogenous variables) models. Though ARMAX is not exactly ARIMA, the author attempted to make use of it to conduct simple forecasting analyses.

According to the documentation by Jean-Philippe Chancelier (2001), the following functions may have potentiality for ARIMA modeling. The function *armax1( )* is used to identify the coefficients of a 1-dimensional ARMA process in the form as follows.

*A(z^-1)y= B(z^-1)u + D(z^-1)sig*e(t)*

Where, u : the input signals;

y: the output signals;

e(t) is a white noise with variance 1;

A(z)= 1+a1*z+...+a_r*z^r; (r=0 => A(z)=1)

B(z)= b0+b1*z+...+b_s z^s (s=-1 => B(z)=0)

D(z)= 1+d1*z+...+d_q*z^q (q=0 => D(z)=1).

The syntax takes the form:

[a,b,d,sig,resid]=armax1(r,s,q,y,u)

where, r,s,q: auto regression and moving average orders;

a: the vector [1,a1,...,a_r]

b: the vector [b0,.....,b_s]

d: the vector [1,d1,....,d_q]

sig is the estimated standard deviation of the noise.

Hence, the function was used to identify an ARMA model by setting $s = -1$. In the result, a represents AR parameters and d represents MA parameters.

Unfortunately, the function has its limitations. As it does not consider differencing and seasonal parameters, it can only be used to fit stationary and non-seasonal data. Even for such data, the result is not ideal. The problem could result from the function definition. The ARMAX model is essentially different from the ARIMA or the ARMA model. To solve this problem, the author recommends creating a time series object to deal with seasonality and defining a new function with other forms of model formula for ARIMA modeling. This is, however, not undertaken in the current work.

3. Making forecasts

Scilab has no prediction functions for ARMAX models. The author attempted to generate forecasts using the functions *armac( )* and *arsimul( )*.

The function *armac( )* is used to describe an ARMAX process. This means that the ARMA model can be defined with the identified parameters, which include a and d from results of the function *armax1( )*. For example, the following form can be used to describe an ARMA process *y_t -0.8*y_{t-1} + 0.2*y_{t-2} = 0.01*e(t-1)*:

myarma=armac([1,-0.8,0.2],[],[1],1,1,sig)

where sig is the standard deviation.

The function *arsimul( )* is used for ARMAX simulation. It takes the following form:

thefit=arsimul(myarma,u)

To generate the forecasts of n-ahead periods, input signals u are necessary. The input was set by default to a vector with the values of zero.

4. Summary

The implementation of a single task for Scilab demonstrates that our approach is feasible and general for scientific software. However, depending on the task at hand, people may have to create more object definitions and write more functions for model fitting or prediction where the package, in our case Scilab, lacks appropriate data structures or computational features. Clearly it is worth selecting the underlying tool carefully to save work. Scilab was not as well suited to our forecasting tasks.

5.2 A Local Implementation for R

Aside from the web-based approach, there are some other possible approaches to adding both forecasting functionality and user interfaces to Free Software. A local solution could be used. We created such an implementation for R on the Windows platform.

1. Analysis and design

Using a local implementation means that the user needs to install the R package on the local computer. Moreover, temporary files are also created locally.

Our GUI is still based on JavaScript-capable browsers. The user does not browse the Internet but use the local computer environment.

The GUI was created using the JavaScript language. The user views an HTML page to access a forecasting tool. On the page, the user is required as before to enter some parameters. Aside from the parameters for loading data, model fitting, making forecasts and outputting results, the user needs to specify the directory paths in which the R package was installed and timeporary files will be saved. After the user clicks the *Go* button, two text files are created on the specified directory on the local computer. One file is an R command file. The other is a batch file that invokes R to run the R command file. Then, the user needs to run the batch file to invoke R and get outputs in text or graphical files saved on the local computer. Figure 5.2 shows the data flow through the system. The circles with numbers inside present the time sequence of data flows.

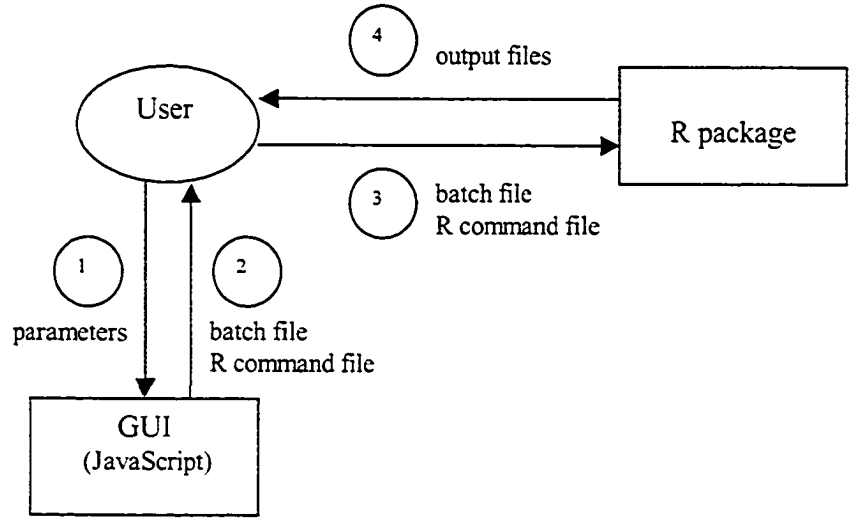

Figure 5.2. The data flows in the local implementation for R (the circles with numbers inside show the time sequence of data flows)

The key issue here is how to generate text files. The JavaScript language has its limitation. Without special plug-ins, JavaScript is essentially a language for client-side applications. It cannot invoke system commands or export files. The IE browser figures the problem out using the ActiveX technology. ActiveX is a loosely defined set of technologies developed by Microsoft for sharing information among different applications (AgMAY Inc., 2002). It is an outgrowth of two other Microsoft technologies called OLE (Object Linking and Embedding) and COM (Component Object Model).

Here an ActiveX object needs to be created to provide file system services (see Appendices E.2 line 165-168). The following example is to create a text file named *example1.bat*.

```
var myobject1=new ActiveXObject("Scripting.FileSystemObject");
var myfile =myobject1.CreateTextFile("example1.bat", true);
myfile.WriteLine("This is a batch file.")
myfile.Close()
```

File system services may result in files in the hard disks modified or lost, so it is necessary to get the privilege to get such services. The user may see a warning window (Figure 5.3) to ask for permission to create text files on his or her computer. So in some lab

environments, students may have to put the HTML programs in particular locations in order to get the privilege to create text files using the Active X technology.

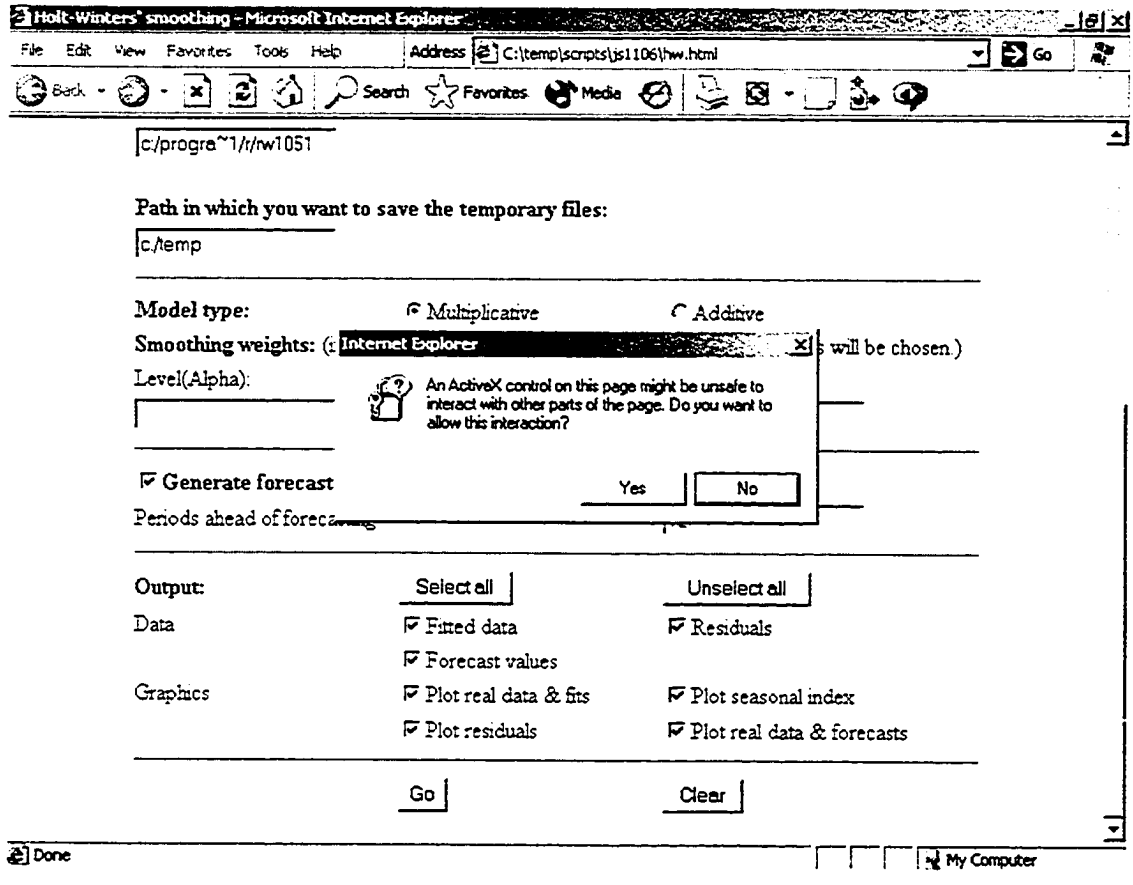

Figure 5.3. The alert window for the privilege permission

This approach is supported only by IE4.0+ and only on Microsoft Windows. Microsoft provides the ActiveX plug-in for NS (Microsoft Corporation, 1996). Note that different versions of the Windows system need different plug-ins.

We also attempted another way to provide supporting information. JavaScript scripts can be put in two sections of the HTML document. Scripts in the body section will execute when the page loads, while scripts in the head section will be executed when called. Thus we can present supporting messages when a page loads, by putting functions in the body section (see Appendices E.2 line 10-17 and line 297). In this way, the user can get important warning before he or she begins to operate in a page form. This may avoid some common errors.

In this local implementation, we present an alert window to give some warning messages. The following screen (Figure 5.4) is what the user sees immediately after the Holt-Winter's method page loads.

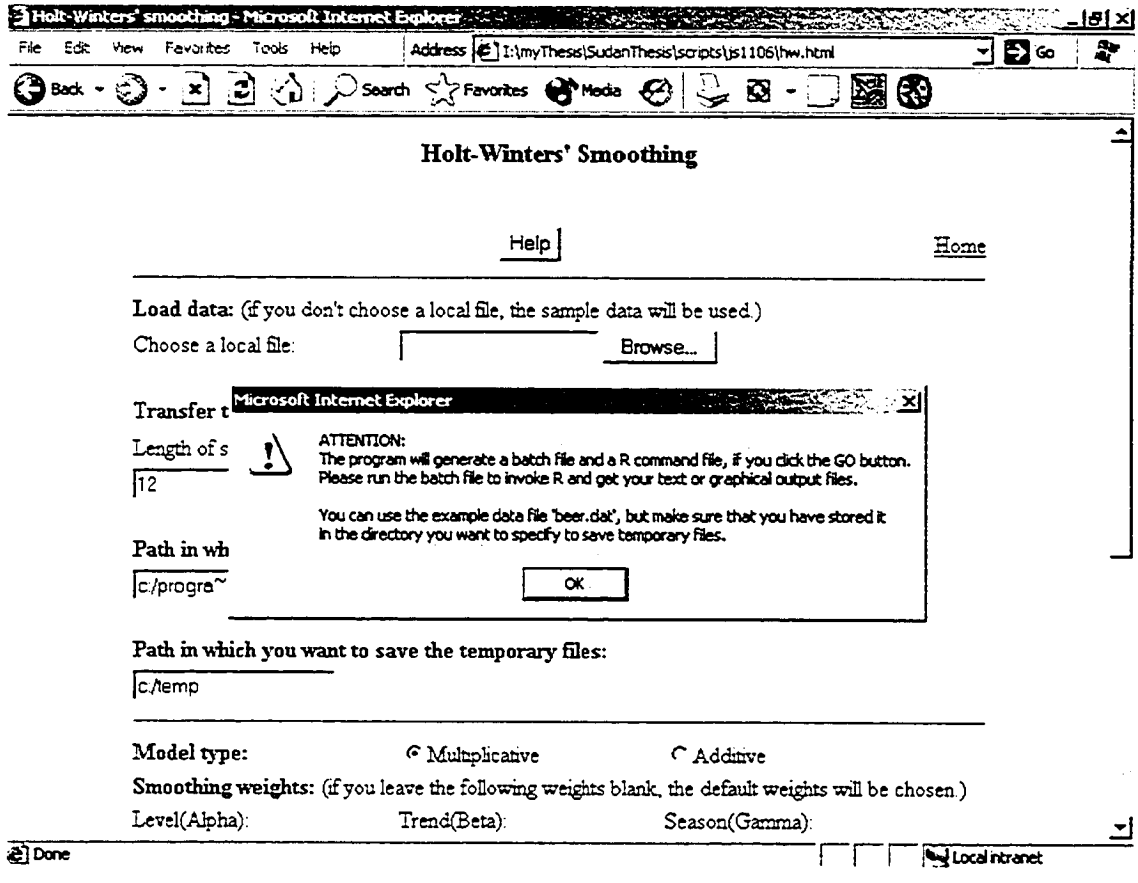

Figure 5.4. The alert window while a page loads

2. Summary

This approach is very efficient in computer run time and in tools used. It uses only two programming languages, JavaScript and R. Generally, it is quick to create a GUI using JavaScript. Many free HTML editors such as 1st Page 2000 and Front Page can provide the previewing function, which makes writing and debugging JavaScript code easy and simple. Thus it is simples than the web-based approach to program.

What users need is R installed locally and a JavaScript-capable browser. So the requirements for users are still simple, though more than that of the web-based approach.

However, the approach has two limitations. First, it cannot be supported by browsers of different types. It depends on the ActiveX technology to create text files. It is only supported by IE. For NS, a plug-in need be installed. And it can only be supported on the Windows operating system. It is always difficult to solve the compatibility problem. In addition, the approach is not very convenient for users. Because batch files have to be run by the user, the approach depends too much on users for its operation. It might not be thought of as a real GUI and the text or graphical results cannot be presented in the GUI.

5.3 A local server approach

The web-based GUIs can also support the use of a local sever. We can install CGI programs into a computer acting as a web server, then start the GUI through localhost. Meanwhile, we want to justify the compatibility of the web-based approach on different platforms. Hence, we moved RForecast and SciForecast from Linux to Windows and found the following issues are worth discussing.

- Use appropriate system command corresponding to the platform

Perl has good compatibility in different platforms, so most of scripts need not to be changed corresponding to the platform. For example, the following path name of the Perl interpreter for Linux can be recognized on Windows as c:\perl\bin\perl.exe.

#! usr/bin/perl

However, system commands are platform specific. Developers should use appropriate system commands, especially the slash symbols in path names. Let's see an example. The following command is used on Linux to invoke R.

System("Rpath/R --vanilla <commands.R >log.txt");

It should be changed as follows to match the Windows platform (see Appendices D.):

System("Rpath\\Rterm --vanilla <commands.R >log.txt");

Note that R for Windows uses *Rterm* to invoke R in batch mode.

- *Adjust the configuration of the web sever*

We used the Abyss Web Server X1 as the web server on Windows, which is freely available (but not open-source, and there may in future be a charge). "Abyss Web Server X1 is a free personal web server available for Windows and Linux operating systems. Despite its tiny

footprint (the executable file size is less than 104 KB); it supports HTTP/1.1, dynamic content generation through CGI/1.1 scripts, Server Side Includes (SSI), custom error pages, and user access control (HTTP authentication/password protected files)." (http://www.aprelium.com/index.html) Note that we could have chosen to use the Apache server, but this is much larger and less easy to configure.

Abyss allows users to set their own server configurations. As far as CGI parameters are concerned, users can specify the Perl interpreter path and CGI paths. They can also set CGI Scripts' Timeout which refers to how long (in seconds) the server waits for a CGI scripts to deliver content before aborting it. The default Timeout is 30 seconds. But a complex CGI program may require more time. This may result in execution errors. We encountered the same problem when our CGI program has to process graphics, which is a time-consuming work. The CGI program could not generate proper outputs. So we need to change the Timeout to larger number to allow CGI programs to process a large number of data.

- *Invoke Scilab on the Windows platform*

We found it difficult to invoke Scilab in batch mode on Windows. R for Windows provides a command *Rterm* to invoke R in batch mode, while Scilab apparently has no such command. The following commands, which can be issued from the "Run" line on the Start Menu or else directly in an MS-DOS window, can invoke Scilab but only with a command-line window, and cannot send a Scilab command file to this command window.

Runscilab.exe –nw <commands.sce>log.txt

Alternatively, we can use

Scilex.exe

These commands make Scilab wait for input from its command line within the window.

This difficulty was resolved by Dr. John Nash. He copied a Scilab command file to file named *.scilab*, and then executed the *scilex.exe* file. This starts a Scilab window and then closes it after execution of the Scilab commands in the *.scilab* file. The defect of this approach is that users can see a Scilab window that is started and closed after several seconds. However, it does accomplish our purpose. We note that this use of a "special file"

is not unique to Scilab. Matlab version 3.5 used a file *startup.m*, while Minitab Student Edition 12 uses *startup.mtb* with invocation by executing the *mtb12st.exe* file.

- *Change working directory when necessary*

When we used the above approach to invoking Scilab, we found that working directory is very important. Developers should be careful to ensure that the current working directory is as same as the directory in which the *.scilab* file is saved. Otherwise, *scilex* cannot find the *.scilab* file. The working directory can be changed using the *chdir* command.

Summary

The local sever approach simplifies the configuration requirement for the operating environment. It makes the web-based CGI approach portable. This is very convenient for users. They do not need to access the Internet, but use the local server. In this way, we resolved the problem that the Active X technology posed.

5.4 Other Approaches

There are yet other possible approaches to facilitating and simplifying the use of R or Scilab. As far as interfacing to these software packages is concerned, technologies such as Tcl/Tk and ASP should be feasible.

1. Tcl/Tk technology

Tcl stands for the Tool Command Language. Created by John Ousterhout, it has been used as a scripting language. "In most cases, Tcl is used in combination with the Tk ("Tool Kit") library, a set of commands and procedures that make it relatively easy to program graphical user interfaces in Tcl." (Pather, undated)

Tcl is interpreted. Tcl scripts can be executed without compiling and linking. Tcl can run on many platforms, such as UNIX, Windows and Macintosh.

The most significant feature of Tcl is its extensibility. If an application requires some functionality not offered by standard Tcl, new Tcl commands can be written in C or Tcl (Pather, undated). It is easy to add new commands to extend the Tcl language. So it can be used to create extension packages for special purposes.

Both R and Scilab can support interfacing with Tcl/Tk. R has a package, *tcltk*, which allows the use of the Tk GUI toolkit. Tk commands can be embedded into the R language. Peter Dalgaard at the University of Copenhagen has discussed the design considerations and the shortcomings of the current implementation (Dalgaard, 2001). Similarly, Scilab has a toolkit, Tcl/Tk interface, created by Bertrand Guiheneuf (Scilab Group, undated).

Therefore, it should be feasible to create a GUI to R or Scilab using Tcl/Tk.

2. ASP technology

ASP stands for Active Server Pages. It is a technology that allows dynamic creation of HTML documents on a web server when they are requested via HTTP (Robinson et al., 2001). It is a Microsoft technology and works with the Internet Information Server (IIS). ASP pages are normal HTML embedded with scripts in the languages such as VBScript, Jscript or C#.

ASP, as a server side technology, is similar in many ways to some other technologies such as CGI. "There are also ASP environments available for other operation systems such as Chilisoft ASP which will run on operation systems such as UNIX." (Garrett, 2001)

Hence, it might be feasible to create a web-based GUI for R or Scilab using ASP technology. Implementation this approach is similar to that of RForecast, but the problem of compatibility in different platforms may be difficult.

Similar to ASP, JSP (JavaServer Pages) might be also feasible for the same task.

Aside from the above technologies, more complex technologies, such as C, C++ or Java, might be feasible. Due to their powerful and sophisticated mechanisms, these programming languages are relatively difficult to learn to use. It may take entry-level users more time to program in such languages. For relatively small projects, they may not be the efficient choices.

Chapter 6 Costs and Benefits

In completing the analysis of our efforts to provide special features or functions to available scientific or statistical Free Software, it is appropriate to consider the costs and benefits of the approach. Some of these can be precisely quantified, such as prices, but most are estimates based on the time and effort expended in creating the examples.

6.1 Prices and Availability

The Free Software community has developed greatly in recent years. Today there are many kinds of Free Software packages from operation systems to tiny toolkits. Hence, some people advocate developing new Free Software only using Free Software. This may be too radical, but it does reflect the vitality of Free Software movement.

In the example implementations, most software required to build the development environment was Free Software. Thus we were able to provide a good example to make use of Free Software. The following table shows the software list. Some of them are not Free Software, but zero-cost.

The list of software packages for the development environment

All packages here are zero-cost.

Software type	Name	Free Software (Yes /No)
Operating system	Linux Red Hat **	Yes
Browser	Netscape Navigator	No
	Internet Explorer	No
	Mozilla	Yes
Web server	Apache	Yes
	Abyss	No
Interpreter	Perl interpreter	Yes
Graphic toolkit	Ghostscript	Yes
	Netpbm	Yes
Statistical or scientific software	R	Yes
	Scilab	Yes
HTML editor	1st page	No
Remote access utility	Secure Shell (SSH)	Yes

** Linux is Free Software, but Red Hat charges for the distribution.

Note that GIF format could be replaced with some patent-free formats, such as JPEG and PNG.

6.2 Time and Effort

The time spent in the example implementations can be classified to two categories: programming time and researching time. The first refers to time spent on programming on computers while the second refers to time spent on finding out what the problems are and how to solve these problems. For the example implementations, it was found that the researching time is about ten times more than the machine time.

Most importantly, we found that the learning effort became easier over researching (Figure 6.1). It took me two months to create the local GUI for R from the start of learning R and JavaScript. But I spent one month on improving it to the web-based GUI, RForecast

and two weeks on creating a similar GUI for Scilab, SciForecast. Finally it took me only one week to move RForecast and SciForecast onto the Windows platform (using a local web server). This is one justification for the thesis. We believe that the lessons we learned in these implementations may also help other developers save time and work.

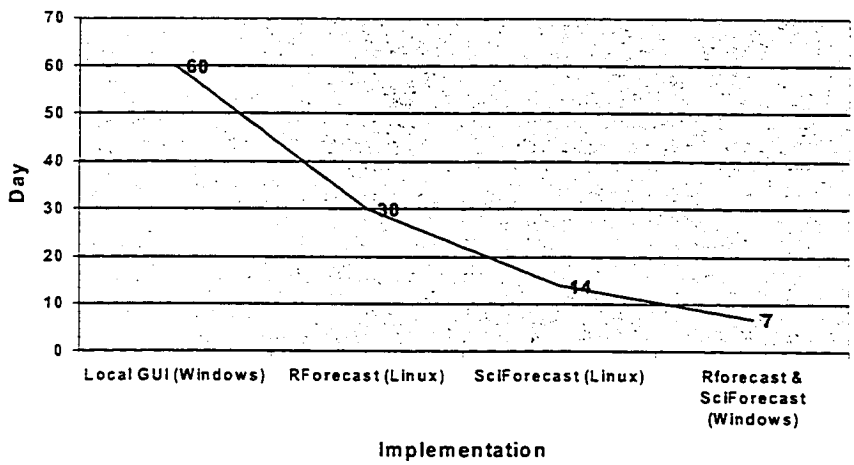

Figure 6.1. The plot of working time over the course of the research

Chapter 7 Conclusions and Recommendations

This thesis made an inquiry into the key issues in adding special purpose functions to free scientific or statistical software. It took R and Scilab as examples and attempted to add forecasting functionality and a GUI to them. The goal of the research is not the implementations themselves but finding out feasible and effective approaches to making full use of Free Software.

Building a GUI for a forecasting task, using a Free Software package as basis, may always be difficult. Taking account of all the blind alleys that we encountered, it might be more economical to simply buy Minitab or an equivalent package. While it may be difficult to provide a fully automatic "recipe", the following approaches we attempted may shorten or smooth the procedure to design and implement such a GUI.

- *Choose an appropriate underlying computational system.* The system should support invocation in batch mode. To save the development time, it would better to have built-in functions that have potential utility for forecasting. R proved to be more suitable than Scilab in this respect. Octave (another free counterpart of Matlab) has no such functions and may be difficult to use for forecasting tasks.
- *Choose an appropriate operating mode and environment.* A GUI can be run on a web server or locally. A portable system, such as the example web-based GUI, can apply to different platforms and web servers, while a local GUI may require restricted configurations. Note that it is possible to use a web-server locally via the localhost facility, offering a compromise.
- *Choose familiar programming languages.* Creating an easy-to-use GUI may require developers' familiarity with programming languages such as Perl and Tcl/Tk. In contrast, one can build a GUI only using JavaScript, which is relatively easy to write and debug.
- *Detect the web browser type and version whenever necessary.* There is always the compatibility problem if use a web browser. At this point, the local GUI using the ActiveX technology is less attractive. To make a GUI work across browsers, special features only supported by a certain browser should be carefully hidden. However, this can result in long and unwieldy programs.

- *Use appropriate commands corresponding to the platform.* Commands in computational languages such as R may defer on different platforms. Invocation commands may also be different between different platforms.
- *Generate text or graphical outputs using available computational tools.* Note that graphic formats produced by our computational engine may have to be converted in order to present graphics in GUI pages. Graphical tools such as Netpbm can be used to perform the conversion. We should not try to force R or Scilab to do "everything".
- *Provide supporting in multiple ways.* We have shown how it is worthwhile to provide such information inline, at the cursor, on the status bar or in a separate window. Note that some web browsers do not support all these approaches.
- *Identify files to avoid user conflicts.* Temporary working files generated on the server side for multiple users need to be kept separate so that an individual user gets the output associated with his or her inputs. Without appropriate identifiers and handling, this is not possible. We found the "process id" to be a good way to identify files.

Implementing a GUI of the type considered in this thesis may cost developers much time to learn both the programming language for the interface and the commands and data structures for the computational system(s). While we have noted that the effort may be great, there are important benefits. The example implementations in this thesis illustrate how to establish working interfaces to available computational tools that simplify the users' access to powerful features of those tools, that is, an operational or production facility for users. They also provide prototypes or models we can modify or adapt to other tasks, that is, a framework for development. It can meet special needs for research, education and business. There is no worry about payment or copyright permission for use of the computational tools, but more importantly, we are not subject to the vendor changing the specifications and requiring us to accept an "upgraded" package that no longer works with our interface. If necessary, we have the source code to work with. In short, the user/developer is in control of his/her own computing.

The research focused only on computational software packages that support invocation in batch mode, so the above approaches used might not apply to other software packages directly. (Note that tools such as ActiveX controls are used to work around such issues.) We

believe that batch invocation simplifies the development of a GUI for tasks such as forecasting.

Recommendations for further research

Making the example interfaces widely applicable to environments in research, education and business still requires a good deal of work. The following modifications and improvements need to be made.

- Developing a spread-sheet window to enter, edit or store data;
- Providing for more statistical functions for forecasting analysis, such as Mean Absolute Deviation (MAD), Mean Squared Deviation (MSD), ACF and Differencing;
- Providing for more types of graphical outputs;
- Providing for automatic optimization for choosing parameters.

While these are important for the functionality of future systems, they are not central to the issues of establishing user-friendly interfaces to Free Software.

References

AgMAY Inc. (2002) *What is ActiveX?* http://www.techiwarehouse.com/Active_X/

Aprelium Home page: http://www.aprelium.com/index.html

Arsham, Hossein (1996) Time Series Analysis and Forecasting Techniques http://obelia.jde.aca.mmu.ac.uk/resdesgn/arsham/opre330Forecast.htm

Banfield, Jeff (1999) *Rweb: Web-based Statistical Analysis.* Journal of Statistical Software,Volume 04 Issue 01 http://www.stat.ucla.edu/journals/jss/v04/i01/

Battilana, Michael C. (2002) *The GIF Controversy: A Software Developer's Perspective* http://cloanto.com/users/mcb/19950127giflzw.html

Bowerman, Bruce L. and O'Connell, Richard T. (1987) *Time series forecasting.* Boston: Duxbury Press.

Box, George E. P. Box and Jenkins, Gwilym M. (1976) *Time series analysis: forecasting and control.* San Francisco: Holden-Day.

Brenner, Steven E. and Aoki, Edwin (1996) *Introduction to CGI/Perl.* New York: M & T books.

Chancelier, Jean-Philippe (2001) *Documentation for Arma modelisation and simulation toolbox* (part of the Scilab documentation from http://www-rocq.inria.fr/scilab/)

Constantine, Larry L. and Lockwood, Lucy A.D. (1999) *Software for use.* New York: ACM Press.

The Computer Technology Documentation Project (CTDP), *CGI Scripting Section Introduction* http://ctdp.tripod.com/independent/web/cgi/

Dalgaard, Peter (2001) *The R-Tcl/Tk Interface* http://www.ci.tuwien.ac.at/Conferences/DSC-2001/Procceddings/Dagaard.pdf

Netscape Communications Corporation (1997) *Sample Code for Form Validation.* DevEdge Online Archive http://developer.netscape.com/docs/examples/javascript/formval/overview.html

Farnum, Nicholas R. and Stanton, LaVerne W. (1989) *Quantitative forecasting methods.* Boston: PWS-KENT.

Feldman, Boris (undated) *DHTML FAQ* http://archive.devx.com/dhtml/dhtmlfaq.asp

Free Software Foundation (FSF) (undated-a)*The Free Software Definition* http://www.gnu.org/philosophy/free-sw.html

Free Software Foundation (FSF) (undated-b) *Categories of Free and Non-Free Software* http://www.gnu.org/philosophy/categories.html

Free Software Foundation (FSF) (1991) *GNU General Public License (1991)* http://www.gnu.org/copyleft/gpl.html

Garrett, Chris (2001) ASP Tutorial http://www.aspalliance.com/chrisg/default.asp?article=83

Hanke, John E. and Reitsh, Arthur G. (1998) *Business Forecast (Six Edition)*. Upper Saddle River, N.J. : Prentice-Hall.

Hornik, Kurt (2002) The R FAQ http://lib/stat.cmu.edu/R/CRAN/doc/FAQ/R-FAQ.html#Introduction

Hyndman (undated), Rob *Time Series Data Library* http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/

Linux Online *What is Linux*, Linux Online Inc., http://www.linux.org/info/index.html

Makridakis, Spayros and Wheelwright, Steven C. (1987) *The Handbook of Forecasting*. New York : John Wiley & Sons, Inc.

Mayhew, Deborah J. (1992) *Principles and Guidelines in Software User Interface Design*. Englewood Cliffs, N.J.: Prentice Hall PTR.

McFarlane, Nigel et al. (1999) *Professional JavaScript*. Chicago: Wrox Press Ltd.

Microsoft Corporation (1996) *What Is ActiveX?* http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/dnaractivex /html/msdn_faq.asp

Nash, John C. and Nash, Mary M. (2001) *Practical forecasting for managers*. London: Arnold Publishers.

Netpbm Home page: http://netpbm.sourceforge.net/

Newbold, Paul and Bos, Theodore (1990) *Introductory Business Forecasting*. Cincinnati, OH : South-Western Publishing Co.

Pather, Shymalan (undated) Introduction to Tcl http://hegel.ittc.ukans.edu/topics/tcltk/tutorial-noplugin/tutorial-1.html

Perens, Bruce (undated) *The Open Source Definition* http://perens.com/Articles/OSD.html

R Development Core Team (2002a) *An Introduction to R (version 1.5.1)*

R Development Core Team (2002b) *R Data Import/Export (version 1.5.1)*

R Development Core Team (2002c) *R Documentation (version 1.5.1)*

R Home Page: *The R Project for Statistical Computing*
http://www.r-project.org/

Robinson, Simon et al. (2001) *Professional C#*, Chicago: Wrox Press ltd.

Scilab Group (undated) *Scilab Reference Manual*
http://www-rocq.inria.fr/scilab/doc/manual/index.html

Scilab Home Page: *Scilab: a FREE CACSD Package by INRIA*
http://www-rocq.inria.fr/scilab/

Shiran, Yehuda and Shiran, Tomer (1998) *Learn Advanced JavaScript Programming.* Worldware Publishing, Inc.

Venables, W. N. and Ripley, B. D. (1994) *Modern applied statistics with S-PLUS.* New York: Springer-Verlag.

Appendices

In this thesis, we implemented RForecast on Linux and Windows, SciForecast on Linux and Windows, and a local CGI for R on Windows. To provide the illustration but keep the thesis modest in length, we included only scripts of three implementations: RForecast (Linux), SciForecast (Windows) and the local GUI for R (Windows).

A. The Configuration and Installation of RForecast (Linux)

The following list shows the detail of the configuration when we developed RForecast.

- *Operating system*: Linux (Red Hat 7.2)
- *Web Server*: Apache 1.3.22
- *Perl interpreter*: Perl 5.6.0
- *Graphical toolkit*: Ghostscript 6.5.1
  Netpbm 9.18
- *Statistical software*: R 1.5.1

To install RForecast locally, please follow the steps as follows:

1. Copy all CGI programs into your CGI path;
2. Copy the example data file (*beer.dat*) and the Help file (help.html) into your HTML path;
3. Install R with the *ts* add-on package;
4. Install ghostscript, pstopnm, ppmtogif, pnmcrop and pnmflip;
5. Change the location path of the Perl interpreter (when necessary);
6. Change path variables in the *varabs.cgi* program.

B. The CGI Programs of RForecast (Linux)

Listing B.1. The RForecast.cgi program

```
1   #!/usr/bin/perl
2   ##RForecast.cgi
3   ##get varables about paths predefined in another file
4   require './varabs.cgi';
5
6   use CGI;
7
8   print ("Content-type:text/html\n\n");
9   print qq|
10  <HTML>
11  <HEAD>
12  <TITLE>RForecast</TITLE>
13  </HEAD>
14  <BODY>
15  <FORM>
16
17
18  <CENTER>
19  <table>
20  <tr>
21    <td width=20%><img src="$imagepath/Rforecast.JPG"></td>
22    <td valign=bottom width=80%><H2 align=center> RForecast: a web-based forecasting
      tool</H2></td>
23  </tr>
24  <tr>
25    <td colspan=2><hr></td>
26  </tr>
27  <tr>
28    <td align=center width=20%><A href="./hw.cgi" onMouseOut="window.status='RForecast'"
      onMouseOver="window.status='Click here to conduct Holt-Winters smoothing analysis';
      return true">Holt-Winters' smoothing</A></td>
29    <td rowspan=3 width=80%>
30      <p>This tool is used to conduct forecasting analysis with time series models.
31       It is based on the free statistical software <a href="http://www.r-
      project.org/">R</a> and provides the following forecasting tools:<br><br>
32       <tools>
33        <li>Holt-Winters' Smoothing</li>
34      <li>Time Series decomposition</li>
35      <li>ARIMA modeling<br></li>
36       </tools>
37       <br>Uers can upload time series from their local computers or use the sample data,
      which is the monthly
38       production (megalitres) of beer in Australia from January 1956 to August 1995.  The
      data can be dowload from
39       <a href="http://www-personal.buseco.monash.edu.au/-hyndman/TSTL">Time Series Data
      Library</a>,
40       provided by Rob Hyndman.<br>
41       <br>The approach to graphic format conversion was introduced by Jeff Banfield in his
      <A href="http://www.math.montana.edu/Rweb/">Rweb</A>.  Rweb is a web-based interface to
      R.
42     </p>
43    </td>
44  </tr>
45  <tr>
46    <td align=center><A href="./tsd.cgi" onMouseOut="window.status='RForecast'"
      onMouseOver="window.status='Click here to conduct Time Series Decomposition analysis';
      return true">Time Series Decomposition</A></td>
47  </tr>
48  <tr>
49    <td align=center><A href="./arima.cgi" onMouseOut="window.status='RForecast'"
      onMouseOver="window.status='Click here to conduct ARIMA modeling analysis'; return
      true">Arima modeling</A></td>
```

```
50  </tr>
51  </table>
52  <br><br>
53  </center>
54  <p>
55  Want to delete temporary files?  Click <A href="./clearup.cgi">here</A> then.
56  </p>
57  |;
58
59  $macnash="http://macnash.admin.uottawa.ca/\~sudan/index.html";
60  $copyright="Copyright\ \&copy;\ 2002 Sudan Wang";
61  $mymail="wsudan\@yahoo.com";
62
63  print qq|
64  <p>
65  Interested in forecasting tools?  Click <A href=$macnash
      onMouseOut="window.status='RForecast'" onMouseOver="window.status='Click here to see
      other implementations'; return true">here</A>
66  to see other implementations for R and <a href="http://www-
      rocq.inria.fr/scilab/">Scilab</a>.
67  </p>
68  <center>
69  <br><hr>
70  <font size=-2>$copyright</font><br>
71  <font size=-2>Any questions or comments please send to <a
      href="mailto:$mymail">$mymail</a></font>
72
73  </CENTER>
74  </FORM>
75  </BODY>
76  </HTML>
77  |;
78  #the end
```

Listing B.2. The varabs.cgi program

```
1   #!/usr/bin/perl
2   ##########################################
3   # This file contains all the variables for RForecast to run properly.
4   # To install the RForecast package locally, users need to change the following
5   # variables to match their own installation.
6   ##########################################
7   #the folder to save R command files
8   $rfilepath="/var/www/html/sudan";
9
10  #the path to call Netpbm
11  $netpbmpath="/usr/bin";
12
13  #the path to link GIF files
14  $imagepath="http://courses.gestion.uottawa.ca/sudan";
15
16  1;
```

Listing B.3 The clearup.cgi program

```
1   #!/usr/bin/perl
2   ##clearup.cgi
3   ##get varables about paths predefined in another file
4   require './varabs.cgi';
5
6   use CGI;
7
8   #delete all temp files
9   my @systemcode=("rm","$rfilepath/*_*.gif");
10  my $runcode=system("@systemcode");
11
12  my @systemcode=("rm","$rfilepath/rlog_*.txt");
13  my $runcode=system("@systemcode");
14
15  my @systemcode=("del","$rfilepath/rfile_*.R");
16  my $runcode=system("@systemcode");
17
18  my @systemcode=("del","$rfilepath/newdata_*.dat");
19  my $runcode=system("@systemcode");
20
21  #show the page
22  print ("Content-type:text/html\n\n");
23  print qq|
24  <HTML>
25  <HEAD>
26  <TITLE>RForecast</TITLE>
27  </HEAD>
28  <BODY>
29  <FORM>
30  <br>
31  <p>All the temporary files saved in the $rfilepath directory have been deleted!</p>
32  <hr>
33  Click <A href="./RForecast.cgi">Here</A> to return the home page.
34  </FORM>
35  </BODY>
36  </HTML>
37  |;
38
39  #the end
```

Listing B.4 The hw.cgi program

```
#!/usr/bin/perl
##hw.cgi
##get varables about paths predefined in another file
require './varabs.cgi';
##get the process id of the Perl program in order to identify temporary files
$processid=$$;

##read in all the variables set by the input form
main:{
  use CGI;
  $query=new CGI;

  if ($query->param('tf') ne "")
  {&processform;}
  else
  {&printform;}
}

#if no parameters sent by the form, then show the form
sub printform
{
  print ("Content-type:text/html\n\n");
  print qq|
  <HTML>
  <HEAD>
  <TITLE> Holt-Winters' smoothing</TITLE>
  <style>
  TABLE {width:80%;}
  span {color:black;}
  </style>
  <script language="Javascript">
  <!--

  function DetectIE()
  {
    //Detect IE version
    IEversion=0
    if (navigator.appVersion.indexOf("MSIE")!=-1)
    {
      verTemp=navigator.appVersion.split("MSIE")
      IEversion=parseFloat(verTemp[1])
    }
  }
  function SureForecast()
  {
   //Detect IE version
   DetectIE()

  //if IE's version is 5.0+, then using special dynamic features
  if (IEversion>=5)
  {
    if (document.hwForm.checkForecast.checked)
    {
     document.hwForm.foreNo.disabled=false
     document.hwForm.foredata.disabled=false
     document.hwForm.realfore.disabled=false
     document.hwForm.all.s1.style.color="black"
     document.hwForm.all.s2.style.color="black"
     document.hwForm.all.s3.style.color="black"
    }
    else
    {
     document.hwForm.foreNo.disabled=true
     document.hwForm.foredata.disabled=true
     document.hwForm.realfore.disabled=true
     document.hwForm.all.s1.style.color="gray"
     document.hwForm.all.s2.style.color="gray"
     document.hwForm.all.s3.style.color="gray"
```

```
69      }
70     }
71    }
72    function checkform()
73    {
74    if (document.hwForm.levelText.value){var theAlpha=document.hwForm.levelText.value}
75    if (document.hwForm.trendText.value){var theGamma=document.hwForm.trendText.value}
76    if (document.hwForm.seasonText.value){var theBeta=document.hwForm.seasonText.value}
77    var a=1
78    var b=1
79    var r=1
80    if (theAlpha<0){a=0}else{if (theAlpha>1){a=0}else {a=1}}
81    if (theBeta<0){b=0}else{if (theBeta>1){b=0}else {b=1}}
82    if (theGamma<0){r=0}else{if (theGamma>1){r=0}else {r=1}}
83    if (a==0)
84    {alert("Invalid weights!  Weights should larger than 0 and less than 1.")}
85    else
86    {
87      if (b==0)
88      {alert("Invalid weights!  Weights should larger than 0 and less than 1.")}
89      else
90      {
91        if (r==0)
92        {alert("Invalid weights!  Weights should larger than 0 and less than 1.")}
93        else
94        {
95        if (!document.hwForm.sealength.value){alert('A season length is required!')}
96        else
97        {
98          if (!document.hwForm.starttime.value){alert('A start time is required!')}
99          else
100         {
101               if (!document.hwForm.startperiod.value){alert('A start period is
    required!')}
102               else{document.hwForm.tf.value="ok"}
103         }
104       }
105      }
106     }
107    }
108   }
109   function theHelp()
110   {
111
        window.open("$imagepath/help.htm","helpWindow","HEIGHT=300,WIDTH=300,menubar,resizabl
    e,scrollbars,status" )
112   }
113   function cursor(text)
114   {
115     //Detect IE version
116     DetectIE()
117
118   //if IE's version is 5.0+, then using special dynamic features
119   if (IEversion>=5)
120   {
121       sprompt.innerHTML=text
122       sprompt.style.color="red"
123      sprompt.style.visibility="visible"
124      sprompt.style.position="absolute"
125      sprompt.style.left=event.clientX-50
126      sprompt.style.top=event.clientY+10
127   }
128   }
129
130   function hidecursor()
131   {
132     //Detect IE version
133     DetectIE()
134
135   //if IE's version is 5.0+, then using special dynamic features
136   if (IEversion>=5)
```

```
137  {sprompt.style.visibility="hidden"}
138  }
139  //-->
140  </script>
141  </HEAD>
142
143  <BODY>
144  <form name="hwForm" action="hw.cgi" enctype="multipart/form-data" method=post>
145
146  <center><h3>Holt-Winters' Smoothing</h3></center>
147  <center>
148  <TABLE>
149    <TR>
150    <td></td>
151    <td align=center><input type="button" value="Help" onclick="theHelp()"></td>
152    <td align=right><A href="./RForecast.cgi" onMouseOut="window.status='RForecst-Winters
     smoothing'" onMouseOver="window.status='Click here to choose other forecasting
     methods'; return true">Home </A></td>
153  </TR>
154    <TR>
155       <td colspan=3><hr></td>
156    </TR>
157  <TR>
158    <TD colspan=3><b>Load data: </b>(if you don't choose a local file, the sample data
     will be used.)</TD>
159  </TR>
160  <TR>
161    <TD>Choose a local file: </TD>
162    <TD colspan=2><INPUT type="file" name="filename"></TD>
163  </TR>
164  <TR>
165    <TD colspan=3><b><BR>Transfer the data to a time series:</b></TD>
166  </TR>
167
168  <TR><span id="sprompt" style="visibility:hidden">WELCOME</span></TR>
169
170  <TR>
171    <TD>Length of season: </TD>
172    <TD>Start time: </TD>
173    <TD>Start period: </TD>
174
175  </TR>
176  <TR>
177    <TD><INPUT type="text" name="sealength" value="12" onBlur="if
     (!document.hwForm.sealength.value){alert('A season length is
     required!');document.hwForm.sealength.focus()}"  onMousemove="cursor('Length of season
     can be 12 months per year, 4 quarters per year, etc.')" onMouseout="hidecursor()"></TD>
178    <TD><INPUT type="text" name="starttime" value="1956" onBlur="if
     (!document.hwForm.starttime.value){alert('A start time is
     required!');document.hwForm.starttime.focus()}" onMousemove="cursor('Start time can be
     year 1956, etc.')" onMouseout="hidecursor()"></TD>
179      <TD><INPUT type="text" name="startperiod" value="1" onBlur="if
     (!document.hwForm.startperiod.value){alert('A start period is
     required!');document.hwForm.startperiod.focus()}" onMousemove="cursor('Start period can
     be first month, third quarter, etc.')" onMouseout="hidecursor()"></TD>
180  </TR>
181    <TR>
182       <td colspan=3><hr></td>
183    </TR>
184  <TR>
185    <TD><b>Model type:</b></TD>
186    <TD><input type="radio" name="modelType" value="multiplicative"
     checked>Multiplicative</TD>
187    <TD><input type="radio" name="modelType" value="additive">Additive</TD>
188  </TR>
189  <TR>
190    <TD colspan=3><b>Smoothing weights: </b>(if you leave the following weights blank,
     the default weights will be chosen.)</TD>
191  </TR>
192  <TR>
193    <TD>Level(Alpha):</TD>
```

```
194     <TD>Trend(Beta):</TD>
195     <TD>Season(Gamma):</TD>
196   </TR>
197   <TR>
198     <TD><input type="text" name="levelText" onmousemove="cursor('The level weight should
      be larger than 0 and less than 1')" onmouseout="hidecursor()"></TD>
199     <TD><input type="text" name="trendText" onmousemove="cursor('The trend weight should
      be larger than 0 and less than 1')" onmouseout="hidecursor()"></TD>
200     <TD><input type="text" name="seasonText" onmousemove="cursor('The season weight
      should be larger than 0 and less than 1')" onmouseout="hidecursor()"></TD>
201   </TR>
202     <tr>
203         <td colspan=3><hr></td>
204     </tr>
205
206   <TR>
207     <TD colspan=3><input type="checkbox" name="checkForecast" checked
      onClick="SureForecast()"><b>Generate forecasts</b></TD>
208   </TR>
209   <TR>
210     <TD colspan=2><span id="s1" >Periods ahead of forecasting:</span>
211     </TD>
212     <TD><input type="text" name="foreNo" ID="foreNo1" value="12" ></TD>
213   </TR>
214     <tr>
215         <td colspan=3><hr></td>
216     </tr>
217   <TR>
218     <TD><b>Output:</b></TD>
219     <TD><input type="button" value="Select all"
      onClick="document.hwForm.fitdata.checked='true';document.hwForm.residata.checked='true'
      ;document.hwForm.foredata.checked='true';document.hwForm.logdata.checked='true';documen
      t.hwForm.realfit.checked='true';document.hwForm.seagraph.checked='true';document.hwForm
      .resigraph.checked='true';document.hwForm.realfore.checked='true';"></TD>
220       <TD><input type="button" value="Unselect all"
      onClick="document.hwForm.fitdata.checked=false;document.hwForm.residata.checked=false;d
      ocument.hwForm.foredata.checked=false;document.hwForm.logdata.checked=false;document.hw
      Form.realfit.checked=false;document.hwForm.seagraph.checked=false;document.hwForm.resig
      raph.checked=false;document.hwForm.realfore.checked=false;"></TD>
221   </TR>
222   <TR>
223     <TD>Data</TD>
224     <TD><input type="checkbox" name="fitdata">Fitted data</TD>
225     <TD><input type="checkbox" name="residata">Residuals</TD>
226   </TR>
227   <TR>
228     <TD></TD>
229     <TD><input type="checkbox" name="foredata" ><span id="s2">Forecast values</span></TD>
230     <TD><input type="checkbox" name="logdata" >R log file</TD>
231   </TR>
232   <TR>
233     <TD>Graphics</TD>
234     <TD><input type="checkbox" name="realfit" checked>Plot real data & fits</TD>
235     <TD><input type="checkbox" name="seagraph">Plot seasonal index</TD>
236   </TR>
237   <TR>
238      <TD></TD>
239      <TD><input type="checkbox" name="resigraph">Plot residuals</TD>
240      <TD><input type="checkbox" name="realfore"><span id="s3">Plot real data &
      forecasts</span></TD>
241   </TR>
242     <tr>
243         <td colspan=3><hr></td>
244     </tr>
245   <tr>
246       <td><input type="hidden" name="tf" value=""></td>
247     <td><input type="button" name="runButton" value=" Go "
      onClick="checkform();submit()">  </td>
248         <td><input type="button" name="clearButton" value=" Clear "
      onClick="reset();document.hwForm.foreNo.value='';document.hwForm.sealength.value='';doc
      ument.hwForm.starttime.value='';document.hwForm.startperiod.value='';"></td>
```

```
249   </tr>
250
251   </TABLE>
252
253   </center>
254   </form>
255   </BODY>
256   </HTML>
257   |;
258 }
259
260 #if got parameters sent from the form, then process the parameters
261 sub processform
262 {
263   #read variables set by the input form
264   $filename=$query->param('filename');
265
266   $sealength=$query->param('sealength');
267   $starttime=$query->param('starttime');
268   $startperiod=$query->param('startperiod');
269
270   $modeltype=$query->param('modelType');
271   $level=$query->param('levelText');
272   if ($level=="") {$level="NULL";}
273   $trend=$query->param('trendText');
274   if ($trend=="") {$trend="NULL";}
275   $season=$query->param('seasonText');
276   if ($season=="") {$season="NULL";}
277   $isfore=$query->param('checkForecast');
278   $foreno=$query->param('foreNo');
279   $fitdata=$query->param('fitdata');
280   $residata=$query->param('residata');
281   $foredata=$query->param('foredata');
282   $logdata=$query->param('logdata');
283   $realfit=$query->param('realfit');
284   $seagraph=$query->param('seagraph');
285   $resigraph=$query->param('resigraph');
286   $realfore=$query->param('realfore');
287   ##print the outputs of the processing
288   print ("Content-type:text/html\n\n");
289   print qq|
290   <HTML>
291   <HEAD><title>Holt-Winters'</title>
292   </HEAD>
293   <BODY>
294   <FORM>
295   <center><h3>Output for Holt-Winters' smoothing</h3></center>
296   <p align="right"><A href="./RForecast.cgi" onMouseOut="window.status='RForecst-Winters
      smoothing'" onMouseOver="window.status='Click here to choose other forecasting
      methods'; return true">Home </A></p>
297   Your outputs are processing now.  It may take several minutes...<BR>
298   |;
299
300   ##read data from the client file ($filename) and write the data to a server file
      ($severfile)
301   $severfile="$rfilepath/newdata_$prodessid.dat";
302   if (-e $filename){
303     if ((-T $filename)&&(-s $filename)){
304       open(DAT,">$severfile");
305       while (read($filename,$buffer,16384)){
306            print DAT $buffer;
307       }
308       close(DAT);
309     }
310     else{
311       ##print the error message
312     print qq|
313       There are some errors when processing your data file $filename.  The reasons may
      be the following: <br>
314       1. The data file is not a text file. <br>
315       2. The data file has a size of zero or unknown characters.<br>
```

```
316       <hr>
317       Please click
318       <A href="./hw.cgi">here </A>to try it again.
319       |;
320     exit();
321      }
322   }
323
324
325
326   ##write R command file
327   $rfile="$rfilepath/rfile_$processid.R";
328   open(DAT,">$rfile");
329   print DAT "##rfile.R\n";
330   print DAT "#Holt-Winters\n";
331   print DAT "library(ts)\n";
332   $sample="$rfilepath/beer.dat";
333   #if the user doesn't choose any local file, then by default use the sample data
      (beer.dat)
334   if ($filename=="")
335   {print DAT "mydata<-read.table(\"$rfilepath\/beer.dat\")\n";}
336   else
337   {print DAT "mydata<-read.table(\"$severfile\")\n";}
338
339   print DAT "n<-nrow(mydata)\n";
340   print DAT "c<-ncol(mydata)\n";
341   print DAT "newc<-c+1\n";
342   print DAT "cycle<-floor(n/$sealength)\n";
343   print DAT "rest<-n-cycle*$sealength\n";
344   print DAT "season<-$sealength\n";
345   print DAT "#convert the data to time series object\n";
346   print DAT "ts.mydata<-
      ts(mydata[,1],start=c($starttime,$startperiod),frequency=$sealength)\n";
347   print DAT "startTime<-$starttime\n";
348   print DAT "startPeriod<-$startperiod\n";
349   print DAT "#fit the data to Holt-Winter model\n";
350   print DAT "hw.mydata<-
      HoltWinters(ts.mydata,alpha=$level,beta=$trend,gamma=$season,seasonal=\"$modeltype\")\n
      ";
351   print DAT "#generate the fitted data\n";
352   print DAT "fit.mydata<-fitted(hw.mydata)\n";
353
354   print DAT "myresult<-
      c(hw.mydata\$seasonal,hw.mydata\$alpha,hw.mydata\$beta,hw.mydata\$gamma,hw.mydata\$SSE)
      \n";
355   print DAT
      "write.table(myresult,file=\"$rfilepath\/myresult_$processid.txt\",quote=FALSE,row.name
      s=c(\"model type:\",\"alpha:\",\"beta:\",\"gamma:\",\"sum of squared
      errors(SSE):\"),col.names=c(\"The final results:\"))\n";
356
357   #if need to make forecast
358   if (defined $isfore)
359   {
360     print DAT "#make forecast\n";
361     print DAT "fhw.mydata<-predict(hw.mydata, n.ahead=$foreno)\n";
362     #output forecast values
363   if (defined $foredata){
364       print DAT "#output forecast values\n";
365     print DAT "ma.fhw<-data.matrix(fhw.mydata)\n";
366       print DAT "ve.fhw<-as.vector(ma.fhw)\n";
367       print DAT
      "write.table(ve.fhw,file=\"$rfilepath\/foredata_$processid.txt\",quote=FALSE,row.names=
      FALSE,col.names=c(\"The forecast values:\"))\n";
368     }
369   #plot real data and forecasts
370     if (defined $realfore){
371       print DAT "#plotting the real data and forecasting values\n";
372       print DAT
      "postscript(file=\"$rfilepath\/realfore_$processid.ps\",width=30,height=30)\n";
373       print DAT "ts.plot(ts.mydata,fhw.mydata,col=c(\"blue\",\"red\"))\n";
```

```
374       print DAT "title(main=\"The plot of real data and forecasts (real data: blue,
    forecasts: red)\")\n";
375     print DAT "dev.off()\n";
376     }
377   }
378   #output fitted data
379   if (defined $fitdata){
380     print DAT "#output fitted data\n";
381     print DAT "ma.fit<-data.matrix(fit.mydata)\n";
382     print DAT "ve.fit<-as.vector(ma.fit)\n";
383     print DAT
    "write.table(ve.fit,file=\"$rfilepath\/fitdata_$processid.txt\",quote=FALSE,row.names=F
    ALSE,col.names=c(\"The fitted data:\"))\n";
384   }
385
386   print DAT "re.mydata<-residuals.HoltWinters(hw.mydata)\n";
387
388   #output residuals
389   if (defined $residata){
390     print DAT "#output residuals\n";
391     print DAT "ma.re<-data.matrix(re.mydata)\n";
392     print DAT "ve.re<-as.vector(ma.re)\n";
393     print DAT
    "write.table(ve.re,file=\"$rfilepath\/residata_$processid.txt\",quote=FALSE,row.names=F
    ALSE,col.names=c(\"The residuals:\"))\n";
394   }
395
396   #plot real data and fits
397   if (defined $realfit){
398     print DAT "#plotting the real data and fits\n";
399     print DAT
    "postscript(file=\"$rfilepath\/realfit_$processid.ps\",width=30,height=30)\n";
400     print DAT "ts.plot(ts.mydata,fit.mydata,col=c(\"blue\",\"red\"))\n";
401     print DAT "title(main=\"The plot of real data and fits (real data: blue, fits:
    red)\")\n";
402   print DAT "dev.off()\n";
403   }
404   #plot residuals
405   if (defined $resigraph){
406     print DAT "#plotting the residuals\n";
407     print DAT
    "postscript(file=\"$rfilepath\/resigraph_$processid.ps\",width=30,height=30)\n";
408     print DAT "plot(re.mydata,col=c(\"blue\"))\n";
409     print DAT "title(main=\"The plot of residuals\")\n";
410     print DAT "dev.off()\n";
411   }
412   #plot seasonal index
413   if (defined $seagraph){
414     print DAT "#plotting the seasonal index\n";
415     print DAT "coef<-hw.mydata\$coefficients\n";
416     print DAT "sea.index<-coef[3:14]\n";
417     print DAT
    "postscript(file=\"$rfilepath\/seagraph_$processid.ps\",width=30,height=30)\n";
418     print DAT "plot(sea.index, type=\"b\")\n";
419     print DAT "title(main=\"The plot of seasonal indices\")\n";
420     print DAT "dev.off()\n";
421   }
422
423   print DAT "\n";
424   close(DAT);
425
426   ##run rfile.txt in batch mode
427   $rlog="$rfilepath/rlog_$processid.txt";
428   system("R --vanilla <$rfile>$rlog");
429
430   print qq|
431   <hr>
432   <b>To save the following results, please use COPY and PASTE.</b><br>
433   |;
434   ##output the result (alpha,beta,gamma,SSE...)
435   $datafile="$rfilepath/myresult_$processid.txt";
```

```
436   if (-e $datafile){
437     open (DAT,$datafile);
438     @data=<DAT>;
439     close(DAT);
440     print "<pre>@data<\/pre>\n";
441   unlink($datafile);
442   }
443
444
445   ##output fits
446   if (defined $fitdata){
447     $datafile="$rfilepath/fitdata_$processid.txt";
448     if (-e $datafile){
449       open (DAT,$datafile);
450       @data=<DAT>;
451       close(DAT);
452     #print "<textarea rows=10 cols=50>@data<\/textarea><br>\n";
453     print "<pre>@data<\/pre>\n";
454     unlink($datafile);
455     }
456   }
457   ##output forecasts
458   if (defined $foredata){
459     $datafile="$rfilepath/foredata_$processid.txt";
460     if (-e $datafile){
461       open (DAT,$datafile);
462       @data=<DAT>;
463       close(DAT);
464     #print "<textarea rows=5 cols=50>@data<\/textarea><br>\n";
465     print "<pre>@data<\/pre>\n";
466     unlink($datafile);
467     }
468   }
469   ##output residuals
470   if (defined $residata){
471     $datafile="$rfilepath/residata_$processid.txt";
472     if (-e $datafile){
473       open (DAT,$datafile);
474       @data=<DAT>;
475       close(DAT);
476     #print "<textarea rows=5 cols=50>@data<\/textarea><br>\n";
477     print "<pre>@data<\/pre>\n";
478     unlink($datafile);
479     }
480   }
481     ##output R log file
482   if (defined $logdata){
483     if (-e $rlog){
484       open (DAT,$rlog);
485       @data=<DAT>;
486       close(DAT);
487     print "R log file:<br>\n";
488     #print "<textarea rows=5 cols=50>@data<\/textarea><br>\n";
489     print "<pre>@data<\/pre>\n";
490     unlink($datafile);
491     }
492   }
493   ##to generate the identifier for ppm file
494   $ppmid=join('',$processid,'001');
495
496   $plotname="$rfilepath/realfit_$processid.ps";
497   if (-e $plotname){
498     ##convert the time series plot from ps format to gif format
499     system("$netpbmpath\/pstopnm -nocrop -yborder 0 -xborder 0 -portrait
      $rfilepath\/realfit_$processid.ps");
500   system("$netpbmpath\/pnmcrop $rfilepath\/realfit_$ppmid.ppm |$netpbmpath\/pnmflip -
      rotate270 |$netpbmpath\/ppmtogif -t rgbi:1/1/1 >$rfilepath\/realfit_$processid.gif");
501
502     ##print the real-fit graph
503   print qq|
504   <br><img src="$imagepath/realfit_$processid.gif"><br>
```

```
505  |;
506
507    #delete temp graphics
508    unlink($plotname);
509  unlink("$rfilepath/realfit_$ppmid.ppm");
510  #unlink("$rfilepath/realfit_$processid.gif");
511  }
512  $plotname="$rfilepath/resigraph_$processid.ps";
513  if (-e $plotname){
514    ##convert the residual plot from ps format to gif format
515    system("$netpbmpath\/pstopnm -nocrop -yborder 0 -xborder 0 -portrait
     $rfilepath\/resigraph_$processid.ps");
516    system("$netpbmpath\/pnmcrop $rfilepath\/resigraph_$ppmid.ppm |$netpbmpath\/pnmflip
     -rotate270 |$netpbmpath\/ppmtogif -t rgbi:1/1/1
     >$rfilepath\/resigraph_$processid.gif");
517
518  ##print the residuals graph
519  print qq|
520  <br><img src="$imagepath/resigraph_$processid.gif"><br>
521  |;
522  #delete temp graphics
523    unlink($plotname);
524  unlink("$rfilepath/resigraph_$ppmid.ppm");
525  #unlink("$rfilepath/resigraph_$processid.gif");
526  }
527  $plotname="$rfilepath/seagraph_$processid.ps";
528  if (-e $plotname){
529    ##convert the seasonal index plot from ps format to gif format
530    system("$netpbmpath\/pstopnm -nocrop -yborder 0 -xborder 0 -portrait
     $rfilepath\/seagraph_$processid.ps");
531    system("$netpbmpath\/pnmcrop $rfilepath\/seagraph_$ppmid.ppm |$netpbmpath\/pnmflip -
     rotate270 |$netpbmpath\/ppmtogif -t rgbi:1/1/1 >$rfilepath\/seagraph_$processid.gif");
532  ##print the seasonals graph
533  print qq|
534  <br><img src="$imagepath/seagraph_$processid.gif"><br>
535  |;
536    #delete temp graphics
537  unlink($plotname);
538  unlink("$rfilepath/seagraph_$ppmid.ppm");
539  #unlink("$rfilepath/seagraph_$processid.gif");
540  }
541  $plotname="$rfilepath/realfore_$processid.ps";
542  if (-e $plotname){
543    ##convert the real-fore plot from ps format to gif format
544    system("$netpbmpath\/pstopnm -nocrop -yborder 0 -xborder 0 -portrait
     $rfilepath\/realfore_$processid.ps");
545    system("$netpbmpath\/pnmcrop $rfilepath\/realfore_$ppmid.ppm |$netpbmpath\/pnmflip -
     rotate270 |$netpbmpath\/ppmtogif -t rgbi:1/1/1 >$rfilepath\/realfore_$processid.gif");
546  ##print the real-fore graph
547  print qq|
548  <br><img src="$imagepath/realfore_$processid.gif"><br>
549  |;
550    #delete temp graphics
551  unlink($plotname);
552  unlink("$rfilepath/realfore_$ppmid.ppm");
553  #unlink("$rfilepath/realfore_$processid.gif");
554  }
555
556  unlink($rlog);
557  unlink($rfile);
558  if (-e $severfile)
559  {unlink($severfile);}
560
561  print qq|
562  </FORM>
563  </BODY>
564  </HTML>
565  |;
566
567  #the end
568 }
```

Listing B.5. The tsd.cgi program

```
1   #!/usr/bin/perl
2   ##tsd.cgi
3   ##get varables about paths predefined in another file
4   require './varabs.cgi';
5   ##get the process id of the Perl program in order to identify temporary files
6   $processid=$$;
7
8   ##read in all the variables set by the input form
9   main:{
10    use CGI;
11    $query=new CGI;
12
13    if ($query->param('tf') ne "")
14    {&processform;}
15    else
16    {&printform;}
17  }
18
19  #if no parameters sent by the form, then show the form
20  sub printform
21  {
22    print ("Content-type:text/html\n\n");
23    print qq|
24    <HTML>
25    <HEAD>
26    <TITLE>Time Series Decomposition</TITLE>
27    <style>
28    TABLE {width:80%;}
29    span {color:black;}
30    </style>
31    <script language="Javascript">
32    <!--
33
34    function DetectIE()
35    {
36      //Detect IE version
37      IEversion=0
38      if (navigator.appVersion.indexOf("MSIE")!=-1)
39      {
40        verTemp=navigator.appVersion.split("MSIE")
41        IEversion=parseFloat(verTemp[1])
42      }
43    }
44    function SureForecast()
45    {
46     //Detect IE version
47     DetectIE()
48
49    //if IE's version is 5.0+, then using special dynamic features
50    if (IEversion>=5)
51    {
52      if (document.tsdForm.checkForecast.checked)
53      {
54       document.tsdForm.foreNo.disabled=false
55       document.tsdForm.foredata.disabled=false
56       document.tsdForm.realfore.disabled=false
57       document.tsdForm.all.s1.style.color="black"
58       document.tsdForm.all.s2.style.color="black"
59       document.tsdForm.all.s3.style.color="black"
60      }
61      else
62      {
63       document.tsdForm.foreNo.disabled=true
64       document.tsdForm.foredata.disabled=true
65       document.tsdForm.realfore.disabled=true
66       document.tsdForm.all.s1.style.color="gray"
```

```
67      document.tsdForm.all.s2.style.color="gray"
68      document.tsdForm.all.s3.style.color="gray"
69     }
70    }
71    }
72    function checkform()
73    {
74    document.tsdForm.tf.value="ok"
75    }
76
77    function theHelp()
78    {
79
      window.open("$imagepath/help.htm","helpWindow","HEIGHT=300,WIDTH=300,menubar,resizable,
      scrollbars,status" )
80    }
81    function cursor(text)
82    {
83      //Detect IE version
84      DetectIE()
85
86    //if IE's version is 5.0+, then using special dynamic features
87    if (IEversion>=5)
88    {
89        sprompt.innerHTML=text
90        sprompt.style.color="red"
91       sprompt.style.visibility="visible"
92       sprompt.style.position="absolute"
93       sprompt.style.left=event.clientX-50
94       sprompt.style.top=event.clientY+10
95    }
96    }
97
98    function hidecursor()
99    {
100     //Detect IE version
101     DetectIE()
102
103   //if IE's version is 5.0+, then using special dynamic features
104   if (IEversion>=5)
105   {sprompt.style.visibility="hidden"}
106   }
107   //-->
108   </script>
109   </HEAD>
110
111   <BODY>
112   <form name="tsdForm" action="tsd.cgi" enctype="multipart/form-data" method=post>
113
114   <center><h3>Time Series Decomposition</h3></center>
115   <center>
116   <TABLE>
117     <TR>
118     <td></td>
119     <td align=center><input type="button" value="Help" onclick="theHelp()"></td>
120     <td align=right><A href="./RForecast.cgi">Home </A></td>
121   </TR>
122     <TR>
123        <td colspan=3><hr></td>
124     </TR>
125   <TR>
126     <TD colspan=3><b>Load data: </b>(if you don't choose a local file, the sample data
      will be used.)</TD>
127   </TR>
128   <TR>
129     <TD>Choose a local file: </TD>
130     <TD colspan=2><INPUT type="file" name="filename"></TD>
131   </TR>
132   <TR>
133     <TD colspan=3><b><BR>Transfer the data to a time series:</b></TD>
134   </TR>
```

```
135
136  <TR><span id="sprompt" style="visibility:hidden">WELCOME</span></TR>
137
138  <TR>
139    <TD>Length of season: </TD>
140    <TD>Start time: </TD>
141    <TD>Start period: </TD>
142
143  </TR>
144  <TR>
145    <TD><INPUT type="text" name="sealength" value="12" onBlur="if
     (!document.tsdForm.sealength.value){alert('A season length is
     required!');document.tsdForm.sealength.focus()}"  onMousemove="cursor('Length of season
     can be 12 months per year, 4 quarters per year, etc.')" onMouseout="hidecursor()"></TD>
146    <TD><INPUT type="text" name="starttime" value="1956" onBlur="if
     (!document.tsdForm.starttime.value){alert('A start time is
     required!');document.tsdForm.starttime.focus()}" onMousemove="cursor('Start time can be
     year 1956, etc.')" onMouseout="hidecursor()"></TD>
147      <TD><INPUT type="text" name="startperiod" value="1" onBlur="if
     (!document.tsdForm.startperiod.value){alert('A start period is
     required!');document.tsdForm.startperiod.focus()}" onMousemove="cursor('Start period
     can be first month, third quarter, etc.')" onMouseout="hidecursor()"></TD>
148  </TR>
149    <TR>
150      <td colspan=3><hr></td>
151    </TR>
152  <TR>
153    <TD><b>Model type:</b></TD>
154    <TD><input type="radio" name="modelType" value="multiplicative"
     checked>Multiplicative</TD>
155    <TD><input type="radio" name="modelType" value="additive">Additive</TD>
156  </TR>
157    <tr>
158      <td colspan=3><hr></td>
159    </tr>
160  <TR>
161    <TD colspan=3><input type="checkbox" name="checkForecast" checked
     onClick="SureForecast()"><b>Generate forecasts</b></TD>
162  </TR>
163  <TR>
164    <TD colspan=2><span id="s1" >Periods ahead of forecasting:</span>
165    </TD>
166    <TD><input type="text" name="foreNo" ID="foreNo1" value="12" ></TD>
167  </TR>
168    <tr>
169      <td colspan=3><hr></td>
170    </tr>
171  <TR>
172    <TD><b>Output:</b></TD>
173    <TD><input type="button" value="Select all"
     onClick="document.tsdForm.fitdata.checked='true';document.tsdForm.residata.checked='tru
     e';document.tsdForm.foredata.checked='true';document.tsdForm.logdata.checked='true';doc
     ument.tsdForm.realfit.checked='true';document.tsdForm.seagraph.checked='true';document.
     tsdForm.resigraph.checked='true';document.tsdForm.realfore.checked='true';"></TD>
174      <TD><input type="button" value="Unselect all"
     onClick="document.tsdForm.fitdata.checked=false;document.tsdForm.residata.checked=false
     ;document.tsdForm.foredata.checked=false;document.tsdForm.logdata.checked=false;documen
     t.tsdForm.realfit.checked=false;document.tsdForm.seagraph.checked=false;document.tsdFor
     m.resigraph.checked=false;document.tsdForm.realfore.checked=false;"></TD>
175  </TR>
176  <TR>
177    <TD>Data</TD>
178    <TD><input type="checkbox" name="fitdata">Fitted data</TD>
179    <TD><input type="checkbox" name="residata">Residuals</TD>
180  </TR>
181  <TR>
182    <TD></TD>
183    <TD><input type="checkbox" name="foredata" ><span id="s2">Forecast values</span></TD>
184    <TD><input type="checkbox" name="logdata" >R log file</TD>
185  </TR>
186  <TR>
```

```
187     <TD>Graphics</TD>
188     <TD><input type="checkbox" name="realfit" checked>Plot real data & fits</TD>
189     <TD><input type="checkbox" name="seagraph">Plot seasonal index</TD>
190   </TR>
191   <TR>
192      <TD></TD>
193      <TD><input type="checkbox" name="resigraph">Plot residuals</TD>
194      <TD><input type="checkbox" name="realfore"><span id="s3">Plot real data &
      forecasts</span></TD>
195   </TR>
196     <tr>
197         <td colspan=3><hr></td>
198     </tr>
199   <tr>
200       <td><input type="hidden" name="tf" value=""></td>
201     <td><input type="button" name="runButton" value=" Go "
      onClick="checkform();submit()">  </td>
202         <td><input type="button" name="clearButton" value=" Clear "
      onClick="reset();document.tsdForm.foreNo.value='';document.tsdForm.sealength.value='';d
      ocument.tsdForm.starttime.value='';document.tsdForm.startperiod.value='';"></td>
203   </tr>
204
205   </TABLE>
206
207   </center>
208   </form>
209   </BODY>
210   </HTML>
211   |;
212 }
213
214 #if got parameters sent from the form, then process the parameters
215 sub processform
216 {
217   #read variables set by the input form
218   $filename=$query->param('filename');
219
220   $sealength=$query->param('sealength');
221   $starttime=$query->param('starttime');
222   $startperiod=$query->param('startperiod');
223
224   $modeltype=$query->param('modelType');
225   $isfore=$query->param('checkForecast');
226   $foreno=$query->param('foreNo');
227   $fitdata=$query->param('fitdata');
228   $residata=$query->param('residata');
229   $foredata=$query->param('foredata');
230   $logdata=$query->param('logdata');
231   $realfit=$query->param('realfit');
232   $seagraph=$query->param('seagraph');
233   $resigraph=$query->param('resigraph');
234   $realfore=$query->param('realfore');
235
236   ##print the outputs of the processing
237   print ("Content-type:text/html\n\n");
238   print qq|
239   <HTML>
240   <HEAD><title>Time Series Decomposition</title>
241   </HEAD>
242   <BODY>
243   <FORM>
244   <center><h3>Output for Time Series Decomposition</h3></center>
245   <p align="right"><A href="./RForecast.cgi" onMouseOut="window.status='RForecst-Time
      Series Decomposition'" onMouseOver="window.status='Click here to choose other
      forecasting methods'; return true">Home </A></p>
246   Your outputs are processing now.  It may take several minutes...<BR>
247   |;
248
249   ##read data from the client file ($filename) and write the data to a server file
      ($severfile)
250   $severfile="$rfilepath/newdata_$prodessid.dat";
```

```
251   if (-e $filename){
252     if ((-T $filename)&&(-s $filename)){
253       open(DAT,">$severfile");
254       while (read($filename,$buffer,16384)){
255            print DAT $buffer;
256       }
257       close(DAT);
258     }
259     else{
260       ##print the error message
261     print qq|
262       There are some errors when processing your data file $filename.  The reasons may
      be the following: <br>
263       1. The data file is not a text file. <br>
264       2. The data file has a size of zero or unknown characters.<br>
265       <hr>
266       Please click
267       <A href="./tsd.cgi">here </A>to try it again.
268       |;
269     exit();
270     }
271   }
272
273
274
275   ##write R command file
276   $rfile="$rfilepath/rfile_$processid.R";
277   open(DAT,">$rfile");
278   print DAT "##rfile.R\n";
279   print DAT "##TSD\n";
280   print DAT "library(ts)\n";
281   $sample="$rfilepath/beer.dat";
282   #if the user doesn't choose any local file, then by default use the sample data
      (beer.dat)
283   if ($filename=="")
284   {print DAT "mydata<-read.table(\"$rfilepath\/beer.dat\")\n";}
285   else
286   {print DAT "mydata<-read.table(\"$severfile\")\n";}
287
288   print DAT "n<-nrow(mydata)\n";
289   print DAT "c<-ncol(mydata)\n";
290   print DAT "newc<-c+1\n";
291   print DAT "cycle<-floor(n/$sealength)\n";
292   print DAT "rest<-n-cycle*$sealength\n";
293   print DAT "season<-$sealength\n";
294   print DAT "#convert the data to time series object\n";
295   print DAT "ts.mydata<-
      ts(mydata[,1],start=c($starttime,$startperiod),frequency=$sealength)\n";
296   print DAT "startTime<-$starttime\n";
297   print DAT "startPeriod<-$startperiod\n";
298   print DAT "#fit the data to TSD model\n";
299   print DAT "mydata[,newc]<-c(1:n)\n";
300   print DAT "lm.mydata<-lm(mydata[,1]~mydata[,newc])\n";
301   print DAT "inte.mydata<-lm.mydata\$coefficients[1]\n";
302   print DAT "slope.mydata<-lm.mydata\$coefficients[2]\n";
303
304   print DAT "tsd.mydata<-decompose(ts.mydata,type=\"$modeltype\")\n";
305   print DAT "index.mydata<-tsd.mydata\$figure\n";
306   print DAT "seasonal.mydata<-rep(index.mydata,times=cycle)\n";
307   print DAT "doSeason<-function(rest,x,y){\n";
308   print DAT "le<-length(x)\n";
309   print DAT "if (rest!=0){\n";
310   print DAT "for (i in 1:rest){\n";
311   print DAT "x[le+i]<-y[i]\n";
312   print DAT "x}}\n";
313   print DAT "else {x}}\n";
314   print DAT "seasonal2.mydata<-doSeason(rest,x=seasonal.mydata,y=index.mydata)\n";
315   print DAT "trend.mydata<-inte.mydata+mydata[,newc]*slope.mydata\n";
316
```

```
317  print DAT "myresult<-
     matrix(c(tsd.mydata\$type,rep(\"\",11),tsd.mydata\$figure[1:12],inte.mydata,rep(\"\",11
     ),slope=c(slope.mydata),rep(\"\",11)),nrow=4,ncol=12,byrow=TRUE)\n";
318  print DAT
     "write.table(myresult,file=\"$rfilepath\/myresult_$processid.txt\",quote=FALSE,row.name
     s=c(\"model type:\",\"seasonal indices:\",\"intercept of the treand equation:\",\"slope
     of the trend equation:\"),col.names=c(\"The final results:\",rep(\"\",11)))\n";
319
320  #check the model type and
321  if ($modeltype == "multiplicative")
322  {print DAT "fit.mydata<-trend.mydata*seasonal2.mydata\n";}
323  else
324  {print DAT "fit.mydata<-trend.mydata+seasonal2.mydata\n";}
325
326  print DAT "#generate the fitted data\n";
327  print DAT "fit.ts<-ts(fit.mydata,start=c(startTime,startPeriod),frequency=season)\n";
328
329  #if need to make forecast
330  if (defined $isfore)
331  {
332    print DAT "#make forecast\n";
333    print DAT "trend.fore<-inte.mydata+c((n+1):(n+$foreno))*slope.mydata\n";
334    print DAT "doForeSea<-function(re,y,fore,sea){\n";
335    print DAT "x<-rep(NA,times=fore)\n";
336    print DAT "for (j in 1:fore)\n";
337    print DAT "{if (re==0)\n";
338    print DAT "{mm<-j%%sea}\n";
339    print DAT "else\n";
340    print DAT "{mm<-(j+re)%%sea}\n";
341    print DAT "if (mm==0)\n";
342    print DAT "{x[j]<-y[sea]}\n";
343    print DAT "else\n";
344    print DAT "{x[j]<-y[mm]}}\n";
345    print DAT "x}\n";
346    print DAT "season.fore<-doForeSea(re=rest,y=index.mydata,fore=$foreno,sea=season)\n";
347
348    #check the model type
349    if ($modeltype == "multiplicative")
350    {print DAT "fore.mydata<-trend.fore*season.fore\n";}
351    else
352    {print DAT "fore.mydata<-trend.fore+season.fore\n";}
353
354     print DAT "doTs<-function(time,cy,re){\n";
355     print DAT "if (re==0){\n";
356     print DAT "mystart<-c((time+cy),1)}\n";
357     print DAT "else{\n";
358     print DAT "mystart<-c((time+cy-1),(re+1))}\n";
359     print DAT "mystart}\n";
360     print DAT "theStart<-doTs(time=startTime,cy=cycle,re=rest)\n";
361     print DAT "fore.ts<-ts(fore.mydata,start=theStart,frequency=season)\n";
362
363    #output forecast values
364  if (defined $foredata){
365    print DAT "#output forecast values\n";
366    print DAT "ma.tsd<-data.matrix(fore.ts)\n";
367    print DAT "ve.tsd<-as.vector(ma.tsd)\n";
368      print DAT
     "write.table(ve.tsd,file=\"$rfilepath\/foredata_$processid.txt\",quote=FALSE,row.names=
     FALSE,col.names=c(\"The forecast values:\"))\n";
369    }
370  #plot real data and forecasts
371    if (defined $realfore){
372      print DAT "#plotting the real data and forecasting values\n";
373      print DAT
     "postscript(file=\"$rfilepath\/realfore_$processid.ps\",width=30,height=30)\n";
374    print DAT "ts.plot(ts.mydata,fore.ts,col=c(\"blue\",\"red\"))\n";
375      print DAT "title(main=\"The plot of real data and forecasts (real data: blue,
     forecasts: red)\")\n";
376      print DAT "dev.off()\n";
377    }
378  }
```

```
#output fitted data
if (defined $fitdata){
  print DAT "#output fitted data\n";
  print DAT "ma.fit<-data.matrix(fit.ts)\n";
  print DAT "ve.fit<-as.vector(ma.fit)\n";
  print DAT
"write.table(ve.fit,file=\"$rfilepath\/fitdata_$processid.txt\",quote=FALSE,row.names=F
ALSE,col.names=c(\"The fitted data:\"))\n";
}
#output residuals
print DAT "re.mydata<-ts.mydata-fit.ts\n";

if (defined $residata){
  print DAT "#output residuals\n";
  print DAT "ma.re<-data.matrix(re.mydata)\n";
  print DAT "ve.re<-as.vector(ma.re)\n";
  print DAT
"write.table(ve.re,file=\"$rfilepath\/residata_$processid.txt\",quote=FALSE,row.names=F
ALSE,col.names=c(\"The residuals:\"))\n";
}
#plot real data and fits
if (defined $realfit){
  print DAT "#plotting the real data and fits\n";
  print DAT
"postscript(file=\"$rfilepath\/realfit_$processid.ps\",width=30,height=30)\n";
  print DAT "ts.plot(ts.mydata,fit.ts,col=c(\"blue\",\"red\"))\n";
  print DAT "title(main=\"The plot of real data and fits (real data: blue, fits:
red)\")\n";
  print DAT "dev.off()\n";
}
#plot residuals
if (defined $resigraph){
  print DAT "#plotting the residuals\n";
  print DAT
"postscript(file=\"$rfilepath\/resigraph_$processid.ps\",width=30,height=30)\n";
  print DAT "plot(re.mydata,col=c(\"blue\"))\n";
  print DAT "title(main=\"The plot of residuals\")\n";
  print DAT "dev.off()\n";
}
#plot seasonal index
if (defined $seagraph){
  print DAT "#plotting the seasonal index\n";
  print DAT "index<-tsd.mydata\$figure\n";
  print DAT
"postscript(file=\"$rfilepath\/seagraph_$processid.ps\",width=100,height=100)\n";
  print DAT "plot(index, type=\"b\")\n";
  print DAT "title(main=\"The plot of seasonal indices\")\n";
  print DAT "dev.off()\n";
}

print DAT "\n";
close(DAT);

##run rfile.txt in batch mode
$rlog="$rfilepath/rlog_$processid.txt";
system("R --vanilla <$rfile>$rlog");

print qq|
<hr>
<b>To save the following results, please use COPY and PASTE.</b><br>
|;

##output the result (model type,seasonal indices...)
$datafile="$rfilepath/myresult_$processid.txt";
if (-e $datafile){
  open (DAT,$datafile);
  @data=<DAT>;
  close(DAT);
  print "<pre>@data<\/pre>\n";
unlink($datafile);
}
```

```
442
443   ##output fits
444   if (defined $fitdata){
445     $datafile="$rfilepath/fitdata_$processid.txt";
446     if (-e $datafile){
447       open (DAT,$datafile);
448       @data=<DAT>;
449       close(DAT);
450     #print "<textarea rows=10 cols=50>@data<\/textarea><br>\n";
451     print "<pre>@data<\/pre>\n";
452     unlink($datafile);
453     }
454   }
455   ##output forecasts
456   if (defined $foredata){
457     $datafile="$rfilepath/foredata_$processid.txt";
458     if (-e $datafile){
459       open (DAT,$datafile);
460       @data=<DAT>;
461       close(DAT);
462     #print "<textarea rows=5 cols=50>@data<\/textarea><br>\n";
463     print "<pre>@data<\/pre>\n";
464     unlink($datafile);
465     }
466   }
467   ##output residuals
468   if (defined $residata){
469     $datafile="$rfilepath/residata_$processid.txt";
470     if (-e $datafile){
471       open (DAT,$datafile);
472       @data=<DAT>;
473       close(DAT);
474     #print "<textarea rows=5 cols=50>@data<\/textarea><br>\n";
475     print "<pre>@data<\/pre>\n";
476     unlink($datafile);
477     }
478   }
479     ##output R log file
480   if (defined $logdata){
481     if (-e $rlog){
482       open (DAT,$rlog);
483       @data=<DAT>;
484       close(DAT);
485     print "R log file:<br>\n";
486     #print "<textarea rows=5 cols=50>@data<\/textarea><br>\n";
487     print "<pre>@data<\/pre>\n";
488     unlink($datafile);
489     }
490   }
491   ##to generate the identifier for ppm file
492   $ppmid=join('',$processid,'001');
493
494   $plotname="$rfilepath/realfit_$processid.ps";
495   if (-e $plotname){
496     ##convert the time series plot from ps format to gif format
497     system("$netpbmpath\/pstopnm -nocrop -yborder 0 -xborder 0 -portrait
      $rfilepath\/realfit_$processid.ps");
498   system("$netpbmpath\/pnmcrop $rfilepath\/realfit_$ppmid.ppm |$netpbmpath\/pnmflip -
      rotate270 |$netpbmpath\/ppmtogif -t rgbi:1/1/1 >$rfilepath\/realfit_$processid.gif");
499
500     ##print the real-fit graph
501   print qq|
502   <br><img src="$imagepath/realfit_$processid.gif"><br>
503   |;
504
505     #delete temp graphics
506     unlink($plotname);
507   unlink("$rfilepath/realfit_$ppmid.ppm");
508   #unlink("$rfilepath/realfit_$processid.gif");
509   }
510   $plotname="$rfilepath/resigraph_$processid.ps";
```

```
511   if (-e $plotname){
512     ##convert the residual plot from ps format to gif format
513     system("$netpbmpath\/pstopnm -nocrop -yborder 0 -xborder 0 -portrait
      $rfilepath\/resigraph_$processid.ps");
514     system("$netpbmpath\/pnmcrop $rfilepath\/resigraph_$ppmid.ppm |$netpbmpath\/pnmflip
      -rotate270 |$netpbmpath\/ppmtogif -t rgbi:1/1/1
      >$rfilepath\/resigraph_$processid.gif");
515
516   ##print the residuals graph
517   print qq|
518   <br><img src="$imagepath/resigraph_$processid.gif"><br>
519   |;
520   #delete temp graphics
521   unlink($plotname);
522   unlink("$rfilepath/resigraph_$ppmid.ppm");
523   #unlink("$rfilepath/resigraph_$processid.gif");
524   }
525   $plotname="$rfilepath/seagraph_$processid.ps";
526   if (-e $plotname){
527     ##convert the seasonal index plot from ps format to gif format
528     system("$netpbmpath\/pstopnm -nocrop -yborder 0 -xborder 0 -portrait
      $rfilepath\/seagraph_$processid.ps");
529     system("$netpbmpath\/pnmcrop $rfilepath\/seagraph_$ppmid.ppm |$netpbmpath\/pnmflip -
      rotate270 |$netpbmpath\/ppmtogif -t rgbi:1/1/1 >$rfilepath\/seagraph_$processid.gif");
530   ##print the seasonals graph
531   print qq|
532   <br><img src="$imagepath/seagraph_$processid.gif"><br>
533   |;
534     #delete temp graphics
535   unlink($plotname);
536   unlink("$rfilepath/seagraph_$ppmid.ppm");
537   #unlink("$rfilepath/seagraph_$processid.gif");
538   }
539   $plotname="$rfilepath/realfore_$processid.ps";
540   if (-e $plotname){
541     ##convert the real-fore plot from ps format to gif format
542     system("$netpbmpath\/pstopnm -nocrop -yborder 0 -xborder 0 -portrait
      $rfilepath\/realfore_$processid.ps");
543     system("$netpbmpath\/pnmcrop $rfilepath\/realfore_$ppmid.ppm |$netpbmpath\/pnmflip -
      rotate270 |$netpbmpath\/ppmtogif -t rgbi:1/1/1 >$rfilepath\/realfore_$processid.gif");
544   ##print the real-fore graph
545   print qq|
546   <br><img src="$imagepath/realfore_$processid.gif"><br>
547   |;
548     #delete temp graphics
549   unlink($plotname);
550   unlink("$rfilepath/realfore_$ppmid.ppm");
551   #unlink("$rfilepath/realfore_$processid.gif");
552   }
553
554   unlink($rlog);
555   unlink($rfile);
556   if (-e $severfile)
557   {unlink($severfile);}
558
559   print qq|
560   </FORM>
561   </BODY>
562   </HTML>
563   |;
564
565   #the end
566 }
```

Listing B.6. The arima.cgi program

```
#!/usr/bin/perl
##arima.cgi
##get varables about paths predefined in another file
require './varabs.cgi';
##get the process id of the Perl program in order to identify temporary files
$processid=$$;

##read in all the variables set by the input form
main:{
  use CGI;
  $query=new CGI;

  if ($query->param('tf') ne "")
  {&processform;}
  else
  {&printform;}
}

#if no parameters sent by the form, then show the form
sub printform
{
  print ("Content-type:text/html\n\n");
  print qq|
  <HTML>
  <HEAD>
  <TITLE>ARIMA</TITLE>
  <style>
  TABLE {width:80%;}
  span {color:black;}
  </style>
  <script language="Javascript">
  <!--

  function DetectIE()
  {
    //Detect IE version
    IEversion=0
    if (navigator.appVersion.indexOf("MSIE")!=-1)
    {
      verTemp=navigator.appVersion.split("MSIE")
      IEversion=parseFloat(verTemp[1])
    }
  }
  function sureSeason()
  {
   //Detect IE version
   DetectIE()

  //if IE's version is 5.0+, then using special dynamic features
  if (IEversion>=5)
  {
      if (arimaForm.IsSeason.checked)
    {
     arimaForm.ARP.disabled=false
     arimaForm.DiffD.disabled=false
     arimaForm.MAQ.disabled=false
     arimaForm.all.s4.style.color="black"
     arimaForm.all.s6.style.color="black"
    }
    else
    {
     arimaForm.ARP.value="0"
     arimaForm.ARP.disabled=true
     arimaForm.DiffD.value="0"
     arimaForm.DiffD.disabled=true
     arimaForm.MAQ.value="0"
     arimaForm.MAQ.disabled=true
```

```
68      arimaForm.all.s4.style.color="gray"
69      arimaForm.all.s6.style.color="gray"
70     }
71    }
72    }
73    function SureForecast()
74    {
75     //Detect IE version
76     DetectIE()
77
78    //if IE's version is 5.0+, then using special dynamic features
79    if (IEversion>=5)
80    {
81      if (document.arimaForm.checkForecast.checked)
82      {
83       document.arimaForm.foreNo.disabled=false
84       document.arimaForm.foredata.disabled=false
85       document.arimaForm.realfore.disabled=false
86       document.arimaForm.all.s1.style.color="black"
87       document.arimaForm.all.s2.style.color="black"
88       document.arimaForm.all.s3.style.color="black"
89      }
90      else
91      {
92       document.arimaForm.foreNo.disabled=true
93       document.arimaForm.foredata.disabled=true
94       document.arimaForm.realfore.disabled=true
95       document.arimaForm.all.s1.style.color="gray"
96       document.arimaForm.all.s2.style.color="gray"
97       document.arimaForm.all.s3.style.color="gray"
98      }
99     }
100   }
101   function checkform()
102   {
103    var ARp=document.arimaForm.ARp.value
104   var ARP=document.arimaForm.ARP.value
105   var Diffd=document.arimaForm.Diffd.value
106   var DiffD=document.arimaForm.DiffD.value
107   var MAq=document.arimaForm.MAq.value
108   var MAQ=document.arimaForm.MAQ.value
109   var p=1
110   if (ARp==""){p=0}else{if (ARp<0){p=0}else{ if (ARp>5){p=0}else{p=1}}}
111   var P=1
112   if (ARP==""){P=0}else{if (ARP<0){P=0}else{ if (ARP>5){P=0}else{P=1}}}
113   var d=1
114   if (Diffd==""){d=0}else{if (Diffd<0){d=0}else{ if (Diffd>5){d=0}else{d=1}}}
115   var D=1
116   if (DiffD==""){D=0}else{if (DiffD<0){D=0}else{ if (DiffD>5){D=0}else{D=1}}}
117   var q=1
118   if (MAq==""){q=0}else{if (MAq<0){q=0}else{ if (MAq>5){q=0}else{q=1}}}
119   var Q=1
120   if (MAQ==""){Q=0}else{if (MAQ<0){Q=0}else{ if (MAQ>5){Q=0}else{Q=1}}}
121
122   if (p==0){alert("Invalid parameters!  AR, Difference and MA should be larger than 0 and
      less than 5.")}
123   else
124   {
125     if (P==0){alert("Invalid parameters!  AR, Difference and MA should be larger than 0
      and less than 5.")}
126     else
127     {
128       if (d==0){alert("Invalid parameters!  AR, Difference and MA should be larger than 0
      and less than 5.")}
129       else
130     {
131       if (D==0){alert("Invalid parameters!  AR, Difference and MA should be larger than 0
      and less than 5.")}
132           else
133       {
```

```
134         if (q==0){alert("Invalid parameters!  AR, Difference and MA should be larger than
    0 and less than 5.")}
135           else
136           {
137             if (Q==0){alert("Invalid parameters!  AR, Difference and MA should be
    larger than 0 and less than 5.")}
138             else
139             {document.arimaForm.tf.value="ok"}
140           }
141         }
142       }
143     }
144    }
145   }
146   function theHelp()
147   {
148
    window.open("$imagepath/help.htm","helpWindow","HEIGHT=300,WIDTH=300,menubar,resizable,
    scrollbars,status" )
149   }
150   function cursor(text)
151   {
152     //Detect IE version
153     DetectIE()
154
155   //if IE's version is 5.0+, then using special dynamic features
156   if (IEversion>=5)
157   {
158       sprompt.innerHTML=text
159       sprompt.style.color="red"
160      sprompt.style.visibility="visible"
161      sprompt.style.position="absolute"
162      sprompt.style.left=event.clientX-50
163      sprompt.style.top=event.clientY+5
164   }
165  }
166
167   function hidecursor()
168   {
169     //Detect IE version
170     DetectIE()
171
172   //if IE's version is 5.0+, then using special dynamic features
173   if (IEversion>=5)
174   {sprompt.style.visibility="hidden"}
175   }
176   //-->
177   </script>
178   </HEAD>
179
180   <BODY>
181   <form name="arimaForm" action="arima.cgi" enctype="multipart/form-data" method=post>
182
183   <center><h3>ARIMA modeling</h3></center>
184   <center>
185   <TABLE>
186     <TR>
187     <td></td>
188     <td align=center><input type="button" value="Help" onclick="theHelp()"></td>
189     <td align=right><A href="./RForecast.cgi">Home </A></td>
190   </TR>
191     <TR>
192        <td colspan=3><hr></td>
193     </TR>
194   <TR>
195     <TD colspan=3><b>Load data: </b>(if you don't choose a local file, the sample data
    will be used.)</TD>
196   </TR>
197   <TR>
198     <TD>Choose a local file: </TD>
199     <TD colspan=2><INPUT type="file" name="filename"></TD>
```

```
200   </TR>
201   <TR>
202     <TD colspan=3><b><BR>Transfer the data to a time series:</b></TD>
203   </TR>
204
205   <TR><span id="sprompt" style="visibility:hidden">WELCOME</span></TR>
206
207   <TR>
208     <TD>Length of season: </TD>
209     <TD>Start time: </TD>
210     <TD>Start period: </TD>
211
212   </TR>
213   <TR>
214     <TD><INPUT type="text" name="sealength" value="12" onBlur="if
      (!document.arimaForm.sealength.value){alert('A season length is
      required!');document.arimaForm.sealength.focus()}"  onMousemove="cursor('Length of
      season can be 12 months per year, 4 quarters per year, etc.')"
      onMouseout="hidecursor()"></TD>
215     <TD><INPUT type="text" name="starttime" value="1956" onBlur="if
      (!document.arimaForm.starttime.value){alert('A start time is
      required!');document.arimaForm.starttime.focus()}" onMousemove="cursor('Start time can
      be year 1956, etc.')" onMouseout="hidecursor()"></TD>
216       <TD><INPUT type="text" name="startperiod" value="1" onBlur="if
      (!document.arimaForm.startperiod.value){alert('A start period is
      required!');document.arimaForm.startperiod.focus()}" onMousemove="cursor('Start period
      can be first month, third quarter, etc.')" onMouseout="hidecursor()"></TD>
217   </TR>
218     <TR>
219        <td colspan=3><hr></td>
220     </TR>
221     <tr>
222      <td></td>
223      <td></td>
224      <td><input type="checkbox" name="IsSeason" checked onClick="sureSeason()"><span
      id="s4" >Fit seasonal model</span></td>
225   </tr>
226     <tr>
227      <td></td>
228      <td>Nonseasonal</td>
229      <td><span id="s6" >Seasonal</span></td>
230   </tr>
231   <TR>
232     <TD>Autoregessive(AR)</TD>
233     <TD><input type="text" name="ARp" onMousemove="cursor('AR should be larger than 0 and
      less than 5.')" onMouseout="hidecursor()"></TD>
234     <TD><input type="text" name="ARP" onMousemove="cursor('AR should be larger than 0 and
      less than 5.')" onMouseout="hidecursor()"></TD>
235   </TR>
236   <TR>
237     <TD>Difference</TD>
238     <TD><input type="text" name="Diffd" onMousemove="cursor('Differenc should be larger
      than 0 and less than 5.')" onMouseout="hidecursor()"></TD>
239     <TD><input type="text" name="DiffD" onMousemove="cursor('Differenc should be larger
      than 0 and less than 5.')" onMouseout="hidecursor()"></TD>
240   </TR>
241   <TR>
242     <TD>Moving average(MA)</TD>
243     <TD><input type="text" name="MAq" onMousemove="cursor('MA should be larger than 0 and
      less than 5.')" onMouseout="hidecursor()"></TD>
244     <TD><input type="text" name="MAQ" onMousemove="cursor('MA should be larger than 0 and
      less than 5.')" onMouseout="hidecursor()"></TD>
245   </TR>
246      <tr>
247         <td colspan=3><input type="checkbox" name="Isconstant" checked>Include constant
      term in model</td>
248     </tr>
249     <tr>
250         <td colspan=3><hr></td>
251     </tr>
252   <TR>
```

```
253     <TD colspan=3><input type="checkbox" name="checkForecast" checked
      onClick="SureForecast()"><b>Generate forecasts</b></TD>
254   </TR>
255   <TR>
256     <TD colspan=2><span id="s1" >Periods ahead of forecasting:</span>
257     </TD>
258     <TD><input type="text" name="foreNo" ID="foreNo1" value="12" ></TD>
259   </TR>
260     <tr>
261         <td colspan=3><hr></td>
262     </tr>
263   <TR>
264     <TD><b>Output:</b></TD>
265     <TD><input type="button" value="Select all"
      onClick="document.arimaForm.residata.checked='true';document.arimaForm.foredata.checked
      ='true';document.arimaForm.logdata.checked='true';document.arimaForm.diaggraph.checked=
      'true';document.arimaForm.realfore.checked='true';"></TD>
266       <TD><input type="button" value="Unselect all"
      onClick="document.arimaForm.residata.checked=false;document.arimaForm.foredata.checked=
      false;document.arimaForm.logdata.checked=false;document.arimaForm.diaggraph.checked=fal
      se;document.arimaForm.realfore.checked=false;"></TD>
267   </TR>
268   <TR>
269     <TD>Data</TD>
270     <TD><input type="checkbox" name="residata">Residuals</TD>
271     <TD><input type="checkbox" name="foredata" ><span id="s2">Forecast values</span></TD>
272   </TR>
273   <TR>
274           <TD></TD>
275     <TD><input type="checkbox" name="logdata" >R log file</TD>
276     <TD></TD>
277   </TR>
278   <TR>
279     <TD>Graphics</TD>
280     <TD><input type="checkbox" name="diaggraph" checked>Plot diag graph</TD>
281         <TD><input type="checkbox" name="realfore" ><span id="s3">Plot real data &
      forecasts</span></TD>
282   </TR>
283     <tr>
284         <td colspan=3><hr></td>
285     </tr>
286   <tr>
287       <td><input type="hidden" name="tf" value=""></td>
288     <td><input type="button" name="runButton" value=" Go "
      onClick="checkform();submit()">  </td>
289         <td><input type="button" name="clearButton" value=" Clear "
      onClick="reset();document.arimaForm.foreNo.value='';document.arimaForm.sealength.value=
      '';document.arimaForm.starttime.value='';document.arimaForm.startperiod.value='';"></td
      >
290   </tr>
291
292   </TABLE>
293
294   </center>
295   </form>
296   </BODY>
297   </HTML>
298   |;
299 }
300
301 #if got parameters sent from the form, then process the parameters
302 sub processform
303 {
304   #read variables set by the input form
305   $filename=$query->param('filename');
306
307   $sealength=$query->param('sealength');
308   $starttime=$query->param('starttime');
309   $startperiod=$query->param('startperiod');
310
311   $isseason=$query->param('IsSeason');
```

```
312  $ARp=$query->param('ARp');
313  $ARP=$query->param('ARP');
314  $Diffd=$query->param('Diffd');
315  $DiffD=$query->param('DiffD');
316  $MAq=$query->param('MAq');
317  $MAQ=$query->param('MAQ');
318  $isconstant=$query->param('Isconstant');
319
320  $isfore=$query->param('checkForecast');
321  $foreno=$query->param('foreNo');
322
323  $residata=$query->param('residata');
324  $foredata=$query->param('foredata');
325  $logdata=$query->param('logdata');
326
327  $diaggraph=$query->param('diaggraph');
328  $realfore=$query->param('realfore');
329  ##print the outputs of the processing
330  print ("Content-type:text/html\n\n");
331  print qq|
332  <HTML>
333  <HEAD><title>ARIMA</title>
334  </HEAD>
335  <BODY>
336  <FORM>
337  <center><h3>Output for ARIMA modeling</h3></center>
338  <p align="right"><A href="./RForecast.cgi" onMouseOut="window.status='RForecst-ARIMA
     modeling'" onMouseOver="window.status='Click here to choose other forecasting methods';
     return true">Home </A></p>
339  Your outputs are processing now.  It may take several minutes...<BR>
340  |;
341
342  ##read data from the client file ($filename) and write the data to a server file
     ($severfile)
343  $severfile="$rfilepath/newdata_$prodessid.dat";
344  if (-e $filename){
345    if ((-T $filename)&&(-s $filename)){
346      open(DAT,">$severfile");
347      while (read($filename,$buffer,16384)){
348           print DAT $buffer;
349      }
350      close(DAT);
351    }
352    else{
353      ##print the error message
354    print qq|
355      There are some errors when processing your data file $filename.  The reasons may
     be the following: <br>
356      1. The data file is not a text file. <br>
357      2. The data file has a size of zero or unknown characters.<br>
358      <hr>
359      Please click
360      <A href="./arima.cgi">here </A>to try it again.
361      |;
362    exit();
363    }
364  }
365
366
367
368  ##write R command file
369  $rfile="$rfilepath/rfile_$processid.R";
370  open(DAT,">$rfile");
371  print DAT "##rfile.R\n";
372  print DAT "#ARIMA\n";
373  print DAT "library(ts)\n";
374  $sample="$rfilepath/beer.dat";
375  #if the user doesn't choose any local file, then by default use the sample data
     (beer.dat)
376  if ($filename=="")
377  {print DAT "mydata<-read.table(\"$rfilepath\/beer.dat\")\n";}
```

```
378   else
379   {print DAT "mydata<-read.table(\"$severfile\")\n";}
380
381   print DAT "n<-nrow(mydata)\n";
382   print DAT "c<-ncol(mydata)\n";
383   print DAT "newc<-c+1\n";
384   print DAT "cycle<-floor(n/$sealength)\n";
385   print DAT "rest<-n-cycle*$sealength\n";
386   print DAT "season<-$sealength\n";
387   print DAT "#convert the data to time series object\n";
388   print DAT "ts.mydata<-
      ts(mydata[,1],start=c($starttime,$startperiod),frequency=$sealength)\n";
389   print DAT "startTime<-$starttime\n";
390   print DAT "startPeriod<-$startperiod\n";
391   print DAT "#fit the data to ARIMA model\n";
392   if (defined $isconstant)
393   {$mymean="TRUE";}
394   else
395   {$mymean="FALSE";}
396
397   if (defined $isseason)
398   {print DAT "arima.mydata<-
      arima(ts.mydata,order=c($ARp,$Diffd,$MAq),seasonal=list(order=c($ARP,$DiffD,$MAQ)),incl
      ude.mean=$mymean)\n";}
399   else
400   {print DAT "arima.mydata<-
      arima(ts.mydata,order=c($ARp,$Diffd,$MAq),include.mean=$mymean)\n";}
401
402
403   print DAT "coef<-length(arima.mydata\$coef)\n";
404   print DAT "thel<-7\n";
405   print DAT "if (coef>8){thel<-(coef-1)}\n";
406   print DAT "myresult<-matrix(c(arima.mydata\$aic,rep(\"\",(thel-
      1)),arima.mydata\$arma[1:7],rep(\"\",(thel-7)),names(arima.mydata\$coef)[1:(coef-
      1)],rep(\"\",(8-coef)),arima.mydata\$coef[1:(coef-1)],rep(\"\",(8-
      coef))),nrow=4,ncol=thel,byrow=TRUE)\n";
407   print DAT
      "write.table(myresult,file=\"$rfilepath\/myresult_$processid.txt\",quote=FALSE,row.name
      s=c(\"AIC:\",\"arima[p,q,P,Q,L,d,D]:\",\"names of coefficients:\",\"values of
      coefficients:\"),col.names=c(\"The final results:\",rep(\"\",6)))\n";
408
409   #if need to make forecast
410   if (defined $isfore)
411   {
412      print DAT "#make forecast\n";
413      print DAT "predict.mydata<-predict(arima.mydata, n.ahead=$foreno)\n";
414    print DAT "fore.mydata<-predict.mydata\$pred\n";
415      #output forecast values
416    if (defined $foredata){
417      print DAT "#output forecast values\n";
418     print DAT "ma.fore<-data.matrix(fore.mydata)\n";
419      print DAT "ve.fore<-as.vector(ma.fore)\n";
420      print DAT
      "write.table(ve.fore,file=\"$rfilepath\/foredata_$processid.txt\",quote=FALSE,row.names
      =FALSE,col.names=c(\"The forecast values:\"))\n";
421      }
422    #plot real data and forecasts
423      if (defined $realfore){
424       print DAT "#plotting the real data and forecasting values\n";
425       print DAT
      "postscript(file=\"$rfilepath\/realfore_$processid.ps\",width=30,height=30)\n";
426       print DAT "ts.plot(ts.mydata,fore.mydata,col=c(\"blue\",\"red\"))\n";
427       print DAT "title(main=\"The plot of real data and forecasts (real data: blue,
      forecasts: red)\")\n";
428       print DAT "dev.off()\n";
429      }
430   }
431
432   print DAT "resi.mydata<-arima.mydata\$residuals\n";
433
434   #output residuals
```

```
435   if (defined $residata){
436     print DAT "#output residuals\n";
437     print DAT "ma.resi<-data.matrix(resi.mydata)\n";
438     print DAT "ve.resi<-as.vector(ma.resi)\n";
439     print DAT
      "write.table(ve.resi,file=\"$rfilepath\/residata_$processid.txt\",quote=FALSE,row.names
      =FALSE,col.names=c(\"The residuals:\"))\n";
440   }
441
442   #plot tsdiag
443   if (defined $diaggraph){
444     print DAT "#plotting tsdiag\n";
445     print DAT
      "postscript(file=\"$rfilepath\/diaggraph_$processid.ps\",width=30,height=30)\n";
446     print DAT "tsdiag(arima.mydata)\n";
447   #  print DAT "title(main=\"The diagnostic plot\")\n";
448     print DAT "dev.off()\n";
449   }
450   #plot residuals
451   if (defined $resigraph){
452     print DAT "#plotting the residuals\n";
453     print DAT
      "postscript(file=\"$rfilepath\/resigraph_$processid.ps\",width=30,height=30)\n";
454     print DAT "plot(resi.mydata,col=c(\"blue\"))\n";
455     print DAT "title(main=\"The plot of residuals\")\n";
456     print DAT "dev.off()\n";
457   }
458
459   print DAT "\n";
460   close(DAT);
461
462   ##run rfile.txt in batch mode
463   $rlog="$rfilepath/rlog_$processid.txt";
464   system("R --vanilla <$rfile>$rlog");
465
466   print qq|
467   <hr>
468   <b>To save the following results, please use COPY and PASTE.</b><br>
469   |;
470
471   ##output the result (model type,seasonal indices...)
472   $datafile="$rfilepath/myresult_$processid.txt";
473   if (-e $datafile){
474     open (DAT,$datafile);
475     @data=<DAT>;
476     close(DAT);
477     print "<pre>@data<\/pre>\n";
478   unlink($datafile);
479   }
480   ##output forecasts
481   if (defined $foredata){
482     $datafile="$rfilepath/foredata_$processid.txt";
483     if (-e $datafile){
484       open (DAT,$datafile);
485       @data=<DAT>;
486       close(DAT);
487     print "<pre>@data<\/pre>\n";
488     unlink($datafile);
489     }
490   }
491   ##output residuals
492   if (defined $residata){
493     $datafile="$rfilepath/residata_$processid.txt";
494     if (-e $datafile){
495       open (DAT,$datafile);
496       @data=<DAT>;
497       close(DAT);
498     #print "<textarea rows=5 cols=50>@data<\/textarea><br>\n";
499     print "<pre>@data<\/pre>\n";
500     unlink($datafile);
501     }
```

```
502   }
503     ##output R log file
504   if (defined $logdata){
505     if (-e $rlog){
506       open (DAT,$rlog);
507       @data=<DAT>;
508       close(DAT);
509     print "R log file:<br>\n";
510     #print "<textarea rows=5 cols=50>@data<\/textarea><br>\n";
511     print "<pre>@data<\/pre>\n";
512     unlink($datafile);
513      }
514   }
515
516
517   ##to generate the identifier for ppm file
518   $ppmid=join('',$processid,'001');
519
520   $plotname="$rfilepath/diaggraph_$processid.ps";
521   if (-e $plotname){
522     ##convert the time series plot from ps format to gif format
523     system("$netpbmpath\/pstopnm -nocrop -yborder 0 -xborder 0 -portrait
      $rfilepath\/diaggraph_$processid.ps");
524     system("$netpbmpath\/pnmcrop $rfilepath\/diaggraph_$ppmid.ppm |$netpbmpath\/pnmflip
      -rotate270 |$netpbmpath\/ppmtogif -t rgbi:1/1/1
      >$rfilepath\/diaggraph_$processid.gif");
525
526   ##print the tsdiag graph
527   print qq|
528   <br><img src="$imagepath/diaggraph_$processid.gif">
529   |;
530   #delete temp graphics
531    unlink($plotname);
532   unlink("$rfilepath/diaggraph_$ppmid.ppm");
533   #unlink("$rfilepath/diaggraph_$processid.gif");
534    }
535
536    $plotname="$rfilepath/realfore_$processid.ps";
537    if (-e $plotname){
538      ##convert the real-fore plot from ps format to gif format
539      system("$netpbmpath\/pstopnm -nocrop -yborder 0 -xborder 0 -portrait
      $rfilepath\/realfore_$processid.ps");
540     system("$netpbmpath\/pnmcrop $rfilepath\/realfore_$ppmid.ppm |$netpbmpath\/pnmflip -
      rotate270 |$netpbmpath\/ppmtogif -t rgbi:1/1/1 >$rfilepath\/realfore_$processid.gif");
541
542   ##print the real-fore graph
543   print qq|
544   <br><img src="$imagepath/realfore_$processid.gif">
545   |;
546     #delete temp graphics
547   unlink($plotname);
548   unlink("$rfilepath/realfore_$ppmid.ppm");
549   #unlink("$rfilepath/realfore_$processid.gif");
550   }
551
552   unlink($rlog);
553   unlink($rfile);
554   if (-e $severfile)
555   {unlink($severfile);}
556
557   print qq|
558   </FORM>
559   </BODY>
560   </HTML>
561   |;
562
563   #the end
564 }
```

C. The Configuration and Installation of SciForecast (Windows)

The following list shows the detail of the configuration when we developed SciForecast.

- *Operating system*: Windows 98
- *Web Server*: Abyss Web Sever X1
- *Perl interpreter*: Perl 5.8.0
- *Statistical software*: Scilab 2.6

To install RForecast locally, please follow the steps as follows:

1. Copy all CGI programs into your CGI path;
2. Copy the example data file (*beer.dat*) and the Help file (help.html) into your HTML path;
3. Install Scilab with the ARMA add-in package;
4. Change the location path of the Perl interpreter (when necessary);
5. Change path variables in the *varabs.cgi* program.

D. The CGI Programs of SciForecast (Windows)

Listing D.1. The SciForecast.cgi program

```
#!c:\perl\bin\perl.exe
##SciForecast.cgi
##get varables about paths predefined in another file
require './scivarabs.cgi';

use CGI;

print ("Content-type:text/html\n\n");
print qq|
<HTML>
<HEAD>
<TITLE>SciForecast</TITLE>
</HEAD>
<BODY>
<FORM>


<CENTER>
<table>
<tr>
  <td width=20%><img src="$imagepath/scilabForecast.JPG"></td>
  <td valign=bottom width=80%><H2 align=center> SciForecast: a web-based forecasting
  tool</H2></td>
</tr>
<tr>
  <td colspan=2><hr></td>
</tr>
<tr>
  <td align=center width=20%><A href="./sciarma.cgi"
  onMouseOut="window.status='SciForecast'" onMouseOver="window.status='Click here to
  conduct ARMA modeling analysis'; return true">ARMA modeling</A></td>
  <td width=80%>
    <p>This tool is used to conduct forecasting analysis with time series models.
     It is based on the free statistical software <a href="http://www-
  rocq.inria.fr/scilab/">Scilab</a> and provides the ARMA modeling tool<br><br>

     <br>Uers can upload time series from their local computers or use the sample data,
  which is the monthly
     production (megalitres) of beer in Australia from January 1956 to August 1995.  The
  data can be dowload from
     <a href="http://www-personal.buseco.monash.edu.au/~hyndman/TSTL">Time Series Data
  Library</a>,
     provided by Rob Hyndman.
  </p>
  </td>
</tr>

</table>
<br><br><br>
</center>
<p>
Want to delete temporary files?  Click <A href="./sciclear.cgi">here</A> then.
</p>
|;


$copyright="Copyright\ \&copy;\ 2002 Sudan Wang";
$mymail="wsudan\@yahoo.com";

print qq|
<p>
Interested in forecasting tools?  Click <A href="$indexpage">here</A>
```

```
56  to see other implementations for R and <a href="http://www-
      rocq.inria.fr/scilab/">Scilab</a>.
57  </p>
58
59  <br><hr>
60  <center>
61  <font size=-2>$copyright</font><br>
62  <font size=-2>Any questions or comments please send to <a
      href="mailto:$mymail">$mymail</a></font>
63
64  </CENTER>
65  </FORM>
66  </BODY>
67  </HTML>
68  |;
69  #the end
```

Listing D.2. The varabs.cgi program

```
1   #!/usr/bin/perl
2
3   # This file contains all the variables for SciForecast to run properly.
4
5   #the folder to save Scilab command files
6   $sfilepath="c:/progra~1/abyssw~1/htdocs/sudan";
7   $sfilepath1="c:\\progra~1\\abyssw~1\\htdocs\\sudan";
8
9   #the path to link GIF files
10  $imagepath="http://localhost/sudan";
11
12  #the path of Scilab
13  $spath="C:/Progra~1/scilab-2.6/bin";
14  $spath1="C:\\Progra~1\\scilab-2.6\\bin";
15
16  #the page which links to RForecast and SciForecast
17  $indexpage="http://localhost/sudan/index.html";
18
19  1;
```

Listing D.3. The sciclear.cgi program

```
#!c:\perl\bin\perl.exe
##sciclear.cgi
##get varables about paths predefined in another file
require './scivarabs.cgi';

use CGI;

#delete all temporary files
my @systemcode=("del","$sfilepath1\\*_*.gif");
my $runcode=system("@systemcode");

my @systemcode=("del","$spath1\\.scilab");
my $runcode=system("@systemcode");

my @systemcode=("del","$sfilepath1\\newdata_*.dat");
my $runcode=system("@systemcode");

#show the page
print ("Content-type:text/html\n\n");
print qq|
<HTML>
<HEAD>
<TITLE>SciForecast</TITLE>
</HEAD>
<BODY>
<FORM>
<br>
<p>All the temporary files saved in the $sfilepath directory have been deleted!</p>
<hr>
Click <A href="./SciForecast.cgi">Here</A> to return the home page.
</FORM>
</BODY>
</HTML>
|;

#the end
```

Listing D.4. The sciarma.cgi program

```
#!c:\perl\bin\perl.exe
##sciarma.cgi
##get varables about paths predefined in another file
require './scivarabs.cgi';
##get the process id of the Perl program in order to identify temporary files
$processid=$$;

##read in all the variables set by the input form
main:{
  use CGI;
  $query=new CGI;

  if ($query->param('tf') ne "")
  {&processform;}
  else
  {&printform;}
}

#if no parameters sent by the form, then show the form
sub printform
{
  print ("Content-type:text/html\n\n");
  print qq|
  <HTML>
  <HEAD>
  <TITLE>ARMA</TITLE>
  <style>
  TABLE {width:80%;}
  span {color:black;}
  </style>
  <script language="Javascript">
  <!--

  function DetectIE()
  {
    //Detect IE version
    IEversion=0
    if (navigator.appVersion.indexOf("MSIE")!=-1)
    {
      verTemp=navigator.appVersion.split("MSIE")
      IEversion=parseFloat(verTemp[1])
    }
  }
  function sureSeason()
  {
   //Detect IE version
   DetectIE()

  //if IE's version is 5.0+, then using special dynamic features
  if (IEversion>=5)
  {
      if (arimaForm.IsSeason.checked)
    {
            arimaForm.ARP.disabled=false
     arimaForm.MAQ.disabled=false
     arimaForm.all.s4.style.color="black"
     arimaForm.all.s6.style.color="black"
    }
    else
    {
       arimaForm.ARP.value="0"
     arimaForm.ARP.disabled=true
     arimaForm.MAQ.value="0"
     arimaForm.MAQ.disabled=true
     arimaForm.all.s4.style.color="gray"
     arimaForm.all.s6.style.color="gray"
    }
  }
```

```
69    }
70    function SureForecast()
71    {
72     //Detect IE version
73     DetectIE()
74
75    //if IE's version is 5.0+, then using special dynamic features
76    if (IEversion>=5)
77    {
78      if (document.arimaForm.checkForecast.checked)
79      {
80              document.arimaForm.foreNo.disabled=false
81       document.arimaForm.foredata.disabled=false
82       document.arimaForm.realfore.disabled=false
83       document.arimaForm.all.s1.style.color="black"
84       document.arimaForm.all.s2.style.color="black"
85       document.arimaForm.all.s3.style.color="black"
86      }
87      else
88      {
89       document.arimaForm.foreNo.disabled=true
90       document.arimaForm.foredata.disabled=true
91       document.arimaForm.realfore.disabled=true
92       document.arimaForm.all.s1.style.color="gray"
93       document.arimaForm.all.s2.style.color="gray"
94       document.arimaForm.all.s3.style.color="gray"
95      }
96    }
97    }
98    function checkform()
99    {
100     var ARp=document.arimaForm.ARp.value
101   var ARP=document.arimaForm.ARP.value
102   var MAq=document.arimaForm.MAq.value
103   var MAQ=document.arimaForm.MAQ.value
104   var p=1
105   if (ARp==""){p=0}else{if (ARp<0){p=0}else{ if (ARp>5){p=0}else{p=1}}}
106   var P=1
107   if (ARP==""){P=0}else{if (ARP<0){P=0}else{ if (ARP>5){P=0}else{P=1}}}
108   var q=1
109   if (MAq==""){q=0}else{if (MAq<0){q=0}else{ if (MAq>5){q=0}else{q=1}}}
110   var Q=1
111   if (MAQ==""){Q=0}else{if (MAQ<0){Q=0}else{ if (MAQ>5){Q=0}else{Q=1}}}
112
113   if (p==0){alert("Invalid parameters!  AR and MA should be larger than 0 and less than
      5.")}
114   else
115   {
116     if (P==0){alert("Invalid parameters!  AR and MA should be larger than 0 and less than
      5.")}
117     else
118     {
119       if (q==0){alert("Invalid parameters!  AR and MA should be larger than 0 and less
      than 5.")}
120     else
121     {
122       if (Q==0){alert("Invalid parameters!  AR and MA should be larger than 0 and less
      than 5.")}
123       else
124       {document.arimaForm.tf.value="ok"}
125     }
126     }
127     }
128   }
129   function theHelp()
130   {
131
      window.open("$imagepath/help.htm","helpWindow","HEIGHT=300,WIDTH=300,menubar,resizable,
      scrollbars,status" )
132   }
133   function cursor(text)
```

```
{
  //Detect IE version
  DetectIE()

//if IE's version is 5.0+, then using special dynamic features
if (IEversion>=5)
{
    sprompt.innerHTML=text
    sprompt.style.color="red"
    sprompt.style.visibility="visible"
    sprompt.style.position="absolute"
    sprompt.style.left=event.clientX-50
    sprompt.style.top=event.clientY+5
}
}

function hidecursor()
{
  //Detect IE version
  DetectIE()

//if IE's version is 5.0+, then using special dynamic features
if (IEversion>=5)
{sprompt.style.visibility="hidden"}
}
//-->
</script>
</HEAD>

<BODY>
<form name="arimaForm" action="sciarma.cgi" enctype="multipart/form-data" method=post>

<center><h3>ARMA modeling</h3></center>
<center>
<TABLE>
  <TR>
  <td></td>
  <td align=center><input type="button" value="Help" onclick="theHelp()"></td>
  <td align=right><A href="./SciForecast.cgi">Home</A></td>
</TR>
  <TR>
     <td colspan=3><hr></td>
  </TR>
<TR>
  <TD colspan=3><b>Load data </b>(if you don't choose a local file, the sample data
will be used.)</TD>
</TR>
<TR>
  <TD>Choose a local file: </TD>
  <TD colspan=2><INPUT type="file" name="filename"></TD>
</TR>
<TR><span id="sprompt" style="visibility:hidden">WELCOME</span></TR>
  <TR>
     <td colspan=3><hr></td>
  </TR>
  <tr>
   <td></td>
   <td></td>
   <td><input type="checkbox" name="IsSeason" checked onClick="sureSeason()"><span
id="s4" >Fit seasonal model</span></td>
</tr>
  <tr>
   <td></td>
   <td>Nonseasonal</td>
   <td><span id="s6" >Seasonal</span></td>
</tr>
<TR>
  <TD>Autoregessive(AR)</TD>
  <TD><input type="text" name="ARp" onMousemove="cursor('AR should be larger than 0 and
less than 5.')" onMouseout="hidecursor()"></TD>
```

```
201     <TD><input type="text" name="ARP" value="0" onMousemove="cursor('AR should be larger
    than 0 and less than 5.')" onMouseout="hidecursor()"></TD>
202   </TR>
203   <TR>
204     <TD>Moving average(MA)</TD>
205     <TD><input type="text" name="MAq" onMousemove="cursor('MA should be larger than 0 and
    less than 5.')" onMouseout="hidecursor()"></TD>
206     <TD><input type="text" name="MAQ" value="0" onMousemove="cursor('MA should be larger
    than 0 and less than 5.')" onMouseout="hidecursor()"></TD>
207   </TR>
208     <tr>
209         <td colspan=3><hr></td>
210     </tr>
211   <TR>
212     <TD colspan=3><input type="checkbox" name="checkForecast" checked
    onClick="SureForecast()"><b>Generate forecasts</b></TD>
213   </TR>
214   <TR>
215     <TD colspan=2><span id="s1" >Periods ahead of forecasting:</span>
216     </TD>
217     <TD><input type="text" name="foreNo" ID="foreNo1" value="12" ></TD>
218   </TR>
219     <tr>
220         <td colspan=3><hr></td>
221     </tr>
222   <TR>
223     <TD><b>Output:</b></TD>
224     <TD><input type="button" value="Select all"
    onClick="document.arimaForm.fitdata.checked='true';document.arimaForm.foredata.checked=
    'true';document.arimaForm.logdata.checked='true';document.arimaForm.realfit.checked='tr
    ue';document.arimaForm.realfore.checked='true';"></TD>
225       <TD><input type="button" value="Unselect all"
    onClick="document.arimaForm.fitdata.checked=false;document.arimaForm.foredata.checked=f
    alse;document.arimaForm.logdata.checked=false;document.arimaForm.realfit.checked=false;
    document.arimaForm.realfore.checked=false;"></TD>
226   </TR>
227   <TR>
228     <TD>Data</TD>
229     <TD><input type="checkbox" name="fitdata">Fitted data</TD>
230     <TD><input type="checkbox" name="foredata" ><span id="s2">Forecast values</span></TD>
231   </TR>
232   <TR>
233           <TD></TD>
234     <TD><input type="checkbox" name="logdata" >Scilab command file</TD>
235     <TD></TD>
236   </TR>
237   <TR>
238     <TD>Graphics</TD>
239     <TD><input type="checkbox" name="realfit" checked>Plot real data & fitted data</TD>
240         <TD><input type="checkbox" name="realfore" ><span id="s3">Plot real data &
    forecasts</span></TD>
241   </TR>
242     <tr>
243         <td colspan=3><hr></td>
244     </tr>
245   <tr>
246       <td><input type="hidden" name="tf" value=""></td>
247     <td><input type="button" name="runButton" value=" Go "
    onClick="checkform();submit()">  </td>
248         <td><input type="button" name="clearButton" value=" Clear "
    onClick="reset();document.arimaForm.foreNo.value='';document.arimaForm.sealength.value=
    '';document.arimaForm.starttime.value='';document.arimaForm.startperiod.value='';"></td
    >
249   </tr>
250
251   </TABLE>
252
253   </center>
254   </form>
255   </BODY>
256   </HTML>
```

```
257   |;
258 }
259
260 #if got parameters sent from the form, then process the parameters
261 sub processform
262 {
263   #read variables set by the input form
264   $filename=$query->param('filename');
265
266   $isseason=$query->param('IsSeason');
267   $ARp=$query->param('ARp');
268   $ARP=$query->param('ARP');
269   $MAq=$query->param('MAq');
270   $MAQ=$query->param('MAQ');
271
272   $isfore=$query->param('checkForecast');
273   $foreno=$query->param('foreNo');
274
275   $fitdata=$query->param('fitdata');
276   $foredata=$query->param('foredata');
277   $logdata=$query->param('logdata');
278   $realfit=$query->param('realfit');
279   $realfore=$query->param('realfore');
280   ##print the outputs of the processing
281   print ("Content-type:text/html\n\n");
282   print qq|
283   <HTML>
284   <HEAD><title>ARMA</title>
285   </HEAD>
286   <BODY>
287   <FORM>
288   <center><h3>Output for ARMA modeling</h3></center>
289   <p align="right"><A href="./SciForecast.cgi" onMouseOut="window.status='SciForecst-
      ARMA modeling'" onMouseOver="window.status='Click here to choose other forecasting
      methods'; return true">Home </A></p>
290   Your outputs are processing now.  It may take several minutes...<BR>
291   |;
292
293   ##read data from the client file ($filename) and write the data to a server file
      ($severfile)
294   $severfile="$sfilepath/newdata_$prodessid.dat";
295   if (-e $filename){
296     if ((-T $filename)&&(-s $filename)){
297       open(DAT,">$severfile");
298       while (read($filename,$buffer,16384)){
299            print DAT $buffer;
300       }
301       close(DAT);
302     }
303     else{
304       ##print the error message
305     print qq|
306       There are some errors when processing your data file $filename.  The reasons may
      be the following: <br>
307       1. The data file is not a text file. <br>
308       2. The data file has a size of zero or unknown characters.<br>
309       <hr>
310       Please click
311       <A href="./sciarma.cgi">here </A>to try it again.
312       |;
313     exit();
314     }
315   }
316
317   ##write Scilab command file
318   open(DAT,">$spath/.scilab");
319
320   print DAT "//scifile.sce\n";
321   print DAT "//ARMA\n";
322   print DAT "CT=1;\n";
323   $sample="$sfilepath/beer12.txt";
```

```
#if the user doesn't choose any local file, then by default use the sample data
(beer.dat)
if ($filename=="")
{print DAT "mydata0=read(\"$sfilepath\/beer12.txt\",-1,1);\n";}
else
{print DAT "mydata0=read(\"$severfile\",-1,1);\n";}

print DAT "mydata=mydata0';\n";
print DAT "u1=zeros(1,length(mydata));\n";

print DAT "//fit the data to ARMAX model\n";
if (defined $isseason)
{print DAT "myarma=armax1($ARp,-1,$MAq,mydata,u1);\n";}
else
{print DAT "myarma=armax1($ARp,-1,$MAq,mydata,u1);\n";}

print DAT "//generate fits\n";
print DAT "mya=myarma(\'a\');\n";
print DAT "myd=myarma(\'d\');\n";
print DAT "mysig=myarma(\'sig\');\n";
print DAT "thearma=armac(mya(1:length(mya)),[],myd(1:length(myd)),1,1,mysig);\n";
print DAT "u2=zeros(1,(length(mydata)+$foreno));\n";
print DAT "thefit=arsimul(thearma,u2);\n";
print DAT "thefitt=thefit';\n";
print DAT "myfit=thefitt(1:length(mydata),1);\n";
#output result information (sig,ar,ma)
print DAT "write(\'$sfilepath\/sig_$processid.txt\',mysig);\n";
print DAT "write(\'$sfilepath\/ar_$processid.txt\',mya);\n";
print DAT "write(\'$sfilepath\/ma_$processid.txt\',myd);\n";

#if need to output fits
if (defined $fitdata)
{
  print DAT "write(\'$sfilepath\/fitdata_$processid.txt\',myfit);\n";
}

#if need to make forecast
if (defined $isfore)
{
    print DAT "myfore=thefitt((length(mydata)+1):(length(mydata)+$foreno),1);\n";
   #output forecast values
 if (defined $foredata){
    print DAT "//output forecast values\n";
  print DAT "write(\'$sfilepath\/foredata_$processid.txt\',myfore);\n";
   }
 #plot real data and forecasts
   if (defined $realfore){
    print DAT "//plotting the real data and forecasting values\n";
    print DAT "driver(\'GIF\');\n";
  print DAT "xinit(\'$sfilepath\/realfore_$processid.gif\');\n";
    print DAT "plot2d([1:length(mydata)],mydata0,2);\n";
  print DAT "plot2d([(length(mydata)+1):(length(mydata)+$foreno)],myfore,5);\n";
    print DAT "xend();\n";
   }
}
#plot real data and forecasts
if (defined $realfit){
  print DAT "//plotting the real data and fitted data\n";
  print DAT "driver(\'GIF\');\n";
print DAT "xinit(\'$sfilepath\/realfit_$processid.gif\');\n";
  print DAT "plot2d([],mydata0,2);\n";
print DAT "plot2d([],myfit,5);\n";
  print DAT "xset(\"font\",2,5);\n";
#print DAT "xtitle([\"The plot of the real data and fitted data\";\"Real data:
blue\";\"Fitted data: red\"],\"Time\");\n";
  #print DAT "titlepage(\"The plot of the real data and fitted data\");\n";
print DAT "xend();\n";
}

print DAT "quit();\n" ;
print DAT "\n";
```

```
393   close(DAT);
394
395   ##run sfile.txt in batch mode
396   #change the working directory
397   chdir $spath;
398   #run Scilab
399   my @systemcode=("$spath1\\scilex.exe");
400   my $runcode=system("@systemcode");
401
402   print qq|
403   <hr>
404   <b>To save the following results, please use COPY and PASTE.</b><br>
405   |;
406
407   ##output the result (AR,MA,sig...)
408   $datafile="$sfilepath/ar_$processid.txt";
409   if (-e $datafile){
410     open (DAT,$datafile);
411     @data=<DAT>;
412     close(DAT);
413   print "The estimated AR:\n";
414     print "<pre>@data<\/pre>\n";
415   unlink($datafile);
416   }
417   $datafile="$sfilepath/ma_$processid.txt";
418   if (-e $datafile){
419     open (DAT,$datafile);
420     @data=<DAT>;
421     close(DAT);
422   print "The estimated MA:\n";
423     print "<pre>@data<\/pre>\n";
424   unlink($datafile);
425   }
426   $datafile="$sfilepath/sig_$processid.txt";
427   if (-e $datafile){
428     open (DAT,$datafile);
429     @data=<DAT>;
430     close(DAT);
431   print "The standard deviation is:\n";
432     print "<pre>@data<\/pre>\n";
433   unlink($datafile);
434   }
435   ##output forecasts
436   if (defined $foredata){
437     $datafile="$sfilepath/foredata_$processid.txt";
438     if (-e $datafile){
439       open (DAT,$datafile);
440       @data=<DAT>;
441       close(DAT);
442     print "The forecast values are:\n";
443     print "<pre>@data<\/pre>\n";
444     unlink($datafile);
445     }
446   }
447   ##output fits
448   if (defined $fitdata){
449     $datafile="$sfilepath/fitdata_$processid.txt";
450     if (-e $datafile){
451       open (DAT,$datafile);
452       @data=<DAT>;
453       close(DAT);
454     print "The fitted data is:\n";
455     print "<pre>@data<\/pre>\n";
456     unlink($datafile);
457     }
458   }
459   ##output Scilab command file
460   if (defined $logdata){
461     if (-e $scifile){
462       open (DAT,$scifile);
463       @data=<DAT>;
```

```
464       close(DAT);
465     print "Scilab command file:<br>\n";
466     print "<pre>@data<\/pre>\n";
467     unlink($datafile);
468      }
469   }
470
471   $plotname="$sfilepath/realfit_$processid.gif";
472   if (-e $plotname){
473   ##print the realfit graph
474   print qq|
475   <br><b><center>The plot of the real data and fitted data (real data: blue; fitted
      data: red)</center></b>
476   <center><img src="$imagepath/realfit_$processid.gif"></center>
477   |;
478   }
479
480   $plotname="$sfilepath/realfore_$processid.gif";
481   if (-e $plotname){
482   ##print the real-fore graph
483   print qq|
484     <br><b><center>The plot of the real data and forecasts (real data: blue; forecasts:
      red)</center></b>
485   <center><img src="$imagepath/realfore_$processid.gif"></center>
486   |;
487
488   }
489
490   if (-e $severfile)
491   {unlink($severfile);}
492
493   print qq|
494   </FORM>
495   </BODY>
496   </HTML>
497   |;
498   #the end
499 }
```

E. The Programs of the Local Implementation for R (Windows)

Installation:

1. Copy all HTML files to appropriate directory (not Network directories);

2. Copy the example data file (beer.data) to the directory you want to save temporary files;

3. Install R with the *ts* add-on package.

Note that you need Internet Explorer 5.0+.

Listing E.1. The localfore.html program

```
<html>
<head>
   <title>Forecasting</title>
</head>
<body>
<form><center>
<h3>Forecasting tools</h3>
<hr>
     <a href="hw.html">Holt-Winters' Smoothing</a><br><br>
     <a href="tsd.html">Time Series Decomposition</a><br><br>
     <a href="arima.html">ARIMA Modeling</a><br>
</center></form>
</body>
</html>
```

Listing E.2. The hw.html program

```
<HTML>
<HEAD>
<TITLE> Holt-Winters' smoothing</TITLE>
<style>
TABLE {width:80%;}
span {color:black;}
</style>
<script language="Javascript">
<!--
function message()
{
alert("ATTENTION:\n"
+"The program will generate a batch file and a R command file, if you click the GO button.\n"
+"Please run the batch file to invoke R and get your text or graphical output files.\n\n"
+"You can use the example data file 'beer.dat', but make sure that you have stored it\n"
+"in the directory you want to specify to save temporary files.")
}
function DetectIE()
{
  //Detect IE version
  IEversion=0
  if (navigator.appVersion.indexOf("MSIE")!=-1)
  {
    verTemp=navigator.appVersion.split("MSIE")
    IEversion=parseFloat(verTemp[1])
  }
}
function SureForecast()
{
   //Detect IE version
   DetectIE()
 //if IE's version is 5.0+, then using special dynamic features
 if (IEversion>=5)
 {
   if (document.hwForm.checkForecast.checked)
   {
    document.hwForm.foreNo.disabled=false
    document.hwForm.foredata.disabled=false
    document.hwForm.realfore.disabled=false
    document.hwForm.all.s1.style.color="black"
    document.hwForm.all.s2.style.color="black"
    document.hwForm.all.s3.style.color="black"
   }
   else
   {
    document.hwForm.foreNo.disabled=true
    document.hwForm.foredata.disabled=true
    document.hwForm.realfore.disabled=true
    document.hwForm.all.s1.style.color="gray"
    document.hwForm.all.s2.style.color="gray"
    document.hwForm.all.s3.style.color="gray"
   }
 }
}
function checkform()
{
   if (document.hwForm.levelText.value){var theAlpha=document.hwForm.levelText.value}
 if (document.hwForm.trendText.value){var theGamma=document.hwForm.trendText.value}
 if (document.hwForm.seasonText.value){var theBeta=document.hwForm.seasonText.value}
 var a=1
 var b=1
 var r=1
```

```
63    if (theAlpha<0){a=0}else{if (theAlpha>1){a=0}else {a=1}}
64    if (theBeta<0){b=0}else{if (theBeta>1){b=0}else {b=1}}
65    if (theGamma<0){r=0}else{if (theGamma>1){r=0}else {r=1}}
66    if (a==0)
67    {alert("Invalid weights!  Weights should larger than 0 and less than 1.")}
68    else
69    {
70      if (b==0)
71      {alert("Invalid weights!  Weights should larger than 0 and less than 1.")}
72      else
73      {
74        if (r==0)
75        {alert("Invalid weights!  Weights should larger than 0 and less than 1.")}

76        else
77      {
78        if (!document.hwForm.sealength.value){alert('A season length is required!')}
79           else
80        {
81          if (!document.hwForm.starttime.value){alert('A start time is required!')}
82             else
83             {
84               if (!document.hwForm.startperiod.value){alert('A start period is
     required!')}
85               //else{document.hwForm.tf.value="ok"}
86             }
87        }
88      }
89      }
90    }
91    }
92    function theHelp()
93    {
94
      helpWindow=window.open("","helpWindow","HEIGHT=300,WIDTH=300,menubar,resizeable,scrollb
      ars,status" )
95      helpWindow.document.open();
96      helpWindow.document.writeln("<html><head><title>my help ");
97      helpWindow.document.writeln("</title></head>");
98      helpWindow.document.writeln("<body><PRE>");
99      helpWindow.document.writeln("This is the help")
100     helpWindow.document.writeln("</PRE></body></html>")
101     helpWindow.document.close()
102   }
103   function cursor(text)
104   {
105       //Detect IE version
106     ..DetectIE()
107   //if IE's version is 5.0+, then using special dynamic features
108   if (IEversion>=5)
109   {
110       sprompt.innerHTML=text
111       sprompt.style.color="red"
112       sprompt.style.visibility="visible"
113       sprompt.style.position="absolute"
114       sprompt.style.left=event.clientX-50
115       sprompt.style.top=event.clientY+10
116   }
117   }
118
119   function hidecursor()
120   {
121       //Detect IE version
122       DetectIE()
123   //if IE's version is 5.0+, then using special dynamic features
124   if (IEversion>=5)
125   {sprompt.style.visibility="hidden"}
126   }
127   function doWinter()
128   {
129   var filename=hwForm.filename.value
```

```
130  var sealength=hwForm.sealength.valu
131  var starttime=hwForm.starttime.value
132  var startperiod=hwForm.startperiod.value
133  var Rpath=hwForm.Rpath.value
134  var rfilepath=hwForm.rfilepath.value
135  var modeltype
136  if (hwForm.modelType[0].checked)
137  {modeltype="multiplicative"}
138  else{modeltype="additive"}
139  if (hwForm.modelType[1].checked)
140  {modeltype="additive"}
141  var level=hwForm.levelText.value
142  if (level=="") {level="NULL"}
143  var trend=hwForm.trendText.value
144  if (trend=="") {trend="NULL"}
145  var season=hwForm.seasonText.value
146  if (season=="") {season="NULL"}
147  var isfore=hwForm.checkForecast.value
148  var foreno=hwForm.foreNo.value
149  var fitdata=hwForm.fitdata.value
150  var residata=hwForm.residata.value
151  var foredata=hwForm.foredata.value
152  var realfit=hwForm.realfit.value
153  var seagraph=hwForm.seagraph.value
154  var resigraph=hwForm.resigraph.value
155  var realfore=hwForm.realfore.value
156  //users need to change the following variables to match their local installation

157  var Rcom=Rpath+"/bin/rterm.exe"
158  var sample=rfilepath+"/beer.dat"
159  //*************************************************************************

160
161  var mybat=rfilepath+"/mybat.bat"
162  var rfile=rfilepath+"/rfile.R"
163  var rlog=rfilepath+"/rlog.txt"
164
165  var fso1=new ActiveXObject("Scripting.FileSystemObject")
166  var myfile = fso1.CreateTextFile(mybat, true)
167  myfile.WriteLine(Rcom+" --vanilla <"+rfile+">rlog")
168  myfile.Close()
169
170  var fso2 = new ActiveXObject("Scripting.FileSystemObject");
171  var file = fso2.CreateTextFile(rfile, true);
172  file.WriteLine("#Holt-Winters")
173  file.WriteLine("library(ts)")
174
175  //if the user doesn't choose any file, then by default use the sample data (beer.dat)

176  if (!filename)
177  {file.WriteLine("mydata<-read.table('"+sample+"')")}
178  else
179  {file.WriteLine("mydata<-read.table('"+filename+"')")}
180  file.WriteLine("n<-nrow(mydata)")
181  file.WriteLine("c<-ncol(mydata)")
182  file.WriteLine("newc<-c+1")
183  file.WriteLine("cycle<-floor(n/"+sealength+")")
184  file.WriteLine("rest<-n-cycle*"+sealength)
185  file.WriteLine("season<-"+sealength)
186  file.WriteLine("#convert the data to time series object")
187  file.WriteLine("ts.mydata<-
     ts(mydata[,1],start=c("+starttime+","+startperiod+"),frequency="+sealength+")")

188  file.WriteLine("startTime<-"+starttime)
189  file.WriteLine("startPeriod<-"+startperiod)
190  file.WriteLine("#fit the data to Holt-Winter model")
191  file.WriteLine("hw.mydata<-
     HoltWinters(ts.mydata,alpha="+level+",beta="+trend+",gamma="+season+",seasonal='"+model
     type+"')")
192  file.WriteLine("#generate the fitted data")
193  file.WriteLine("fit.mydata<-fitted(hw.mydata)")
```

```
194
195  var datafile1=""
196  var datafile2=""
197  var datafile2=""
198  var datafile4=""
199  var plotfile1=""
200  var plotfile2=""
201  var plotfile3=""
202  var plotfile4=""
203  //if need to make forecast
204  if (isfore)
205  {
206    file.WriteLine("#make forecast")
207    file.WriteLine("fhw.mydata<-predict(hw.mydata, n.ahead="+foreno+")")

208    //output forecast values
209   if (foredata){
210       file.WriteLine("#output forecast values")
211     file.WriteLine("ma.fhw<-data.matrix(fhw.mydata)")
212       file.WriteLine("ve.fhw<-as.vector(ma.fhw)")
213

     file.WriteLine("write.table(ve.fhw,file='"+rfilepath+"/foredata.txt',quote=FALSE,row.na
     mes=FALSE,col.names=c('The forecast values:'))")
214      var datafile1=rfilepath+"/foredata.txt"
215   }
216   //plot real data and forecasts
217    if (realfore){
218      file.WriteLine("#plotting the real data and forecasting values")
219      file.WriteLine("bmp(file='"+rfilepath+"/realfore.bmp',width=300,height=300)")

220      file.WriteLine("ts.plot(ts.mydata,fhw.mydata,col=c('blue','red'))")

221      file.WriteLine("title(main='The plot of real data and forecasts (real data: blue,
     forecasts: red)')")
222     file.WriteLine("dev.off()")
223     var plotfile1=rfilepath+"/realfore.bmp
224    }
225  }
226
227  //output fitted data
228  if (fitdata){
229    file.WriteLine("#output fitted data")
230    file.WriteLine("ma.fit<-data.matrix(fit.mydata)")
231    file.WriteLine("ve.fit<-as.vector(ma.fit)")
232

     file.WriteLine("write.table(ve.fit,file='"+rfilepath+"/fitdata.txt',quote=FALSE,row.nam
     es=FALSE,col.names=c('The fitted data:'))")
233   var datafile2=rfilepath+"/fitdata.txt"
234  }
235
236  file.WriteLine("re.mydata<-residuals.HoltWinters(hw.mydata)")
237
238  //output residuals
239  if (residata){
240    file.WriteLine("#output residuals")
241    file.WriteLine("ma.re<-data.matrix(re.mydata)")
242    file.WriteLine("ve.re<-as.vector(ma.re)"
243

     file.WriteLine("write.table(ve.re,file='"+rfilepath+"/residata.txt',quote=FALSE,row.nam
     es=FALSE,col.names=c('The residuals:'))")
244   var datafile3=rfilepath+"/residata.txt"
245  }
246
247  //plot real data and fits
248  if (realfit){
249    file.WriteLine("#plotting the real data and fits")
250    file.WriteLine("bmp(file='"+rfilepath+"/realfit.bmp',width=300,height=300)")

251    file.WriteLine("ts.plot(ts.mydata,fit.mydata,col=c('blue','red'))")
252    file.WriteLine("title(main='The plot of real data and fits (real data: blue, fits:
     red)')")
```

```
253    file.WriteLine("dev.off()")
254    var plotfile2=rfilepath+"/realfit.bmp"
255   }
256   //plot residuals
257   if (resigraph){
258    file.WriteLine("#plotting the residuals")
259    file.WriteLine("bmp(file='"+rfilepath+"/resigraph.bmp',width=300,height=300)")

260    file.WriteLine("plot(re.mydata,col=c('blue'))")
261    file.WriteLine("title(main='The plot of residuals')")
262    file.WriteLine("dev.off()")
263    var plotfile3=rfilepath+"/resigraph.bmp"
264   }
265   //plot seasonal index
266   if (seagraph){
267    file.WriteLine("#plotting the seasonal index")
268    file.WriteLine("coef<-hw.mydata$coefficients")
269    file.WriteLine("sea.index<-coef[3:14]")
270    file.WriteLine("bmp(file='"+rfilepath+"/seagraph.bmp',width=300,height=300)")

271    file.WriteLine("plot(sea.index, type='b')")
272    file.WriteLine("title(main='The plot of seasonal indices')")
273    file.WriteLine("dev.off()")
274   var plotfile4=rfilepath+"/seagraph.bmp"
275   }
276
277   file.WriteLine("myresult<-
      c(hw.mydata$seasonal,hw.mydata$alpha,hw.mydata$beta,hw.mydata$gamma,hw.mydata$SSE)")

278
      file.WriteLine("write.table(myresult,file='"+rfilepath+"/myresult.txt',quote=FALSE,row.
      names=c('model type:','alpha:','beta:','gamma:','sum of squared
      errors(SSE):'),col.names=c('The final results:'))")
279   var datafile4=rfilepath+"/myresult.txt"
280   file.WriteLine("")
281   file.Close()
282
283   var datafile=datafile1+" "+datafile2+" "+datafile3+" "+datafile4
284   var plotfile=plotfile1+" "+plotfile2+" "+plotfile3+" "+plotfile4
285   alert("Now please run the batch file '"+mybat+"' to invoke R! \n\n"
286   +"Then you will get the following files in c:/temp: \n"
287   +"1. The file which performs forecasting and plotting: "+rfile+" ;\n"
288   +"2. The R log file:"+rlog+" ;\n"
289   +"3. The plots include: "+plotfile+" ;\n"
290   +"4. The output data files include: "+datafile+" .\n\n"
291   +"ATTENTION: please move or rename the above files for the future use, otherwise next
      operation will overwrite them!!!")
292   }
293   //-->
294   </script>
295   </HEAD>
296
297   <BODY onload="message()">
298   <form name="hwForm" action="hw.cgi" enctype="multipart/form-data" method=post>

299   <center><h3>Holt-Winters' Smoothing</h3></center>
300   <center>
301   <TABLE>
302     <TR>
303      <td></td>
304      <td align=center><input type="button" value="Help" onclick="theHelp()"></td>

305      <td align=right><A href="localfore.html">Home </A></td>
306    </TR>
307     <TR>
308        <td colspan=3><hr></td>
309     </TR>
310    <TR>
311     <TD colspan=3><b>Load data: </b>(if you don't choose a local file, the sample data
      will be used.)</TD>
312    </TR>
```

```
313  <TR>
314    <TD>Choose a local file: </TD>
315    <TD colspan=2><INPUT type="file" name="filename"></TD>
316  </TR>
317  <TR>
318    <TD colspan=3><b><BR>Transfer the data to a time series:</b></TD>
319  </TR>
320  <TR><span id="sprompt" style="visibility:hidden">WELCOME</span></TR>
321  <TR>
322    <TD>Length of season: </TD>
323    <TD>Start time: </TD>
324    <TD>Start period: </TD>
325  </TR>
326  <TR>
327    <TD><INPUT type="text" name="sealength" value="12" onBlur="if
     (!document.hwForm.sealength.value){alert('A season length is
     required!');document.hwForm.sealength.focus()}"  onMousemove="cursor('Length of season
     can be 12 months per year, 4 quarters per year, etc.')" onMouseout="hidecursor()"></TD>

328    <TD><INPUT type="text" name="starttime" value="1956" onBlur="if
     (!document.hwForm.starttime.value){alert('A start time is
     required!');document.hwForm.starttime.focus()}" onMousemove="cursor('Start time can be
     year 1956, etc.')" onMouseout="hidecursor()"></TD>
329      <TD><INPUT type="text" name="startperiod" value="1" onBlur="if
     (!document.hwForm.startperiod.value){alert('A start period is
     required!');document.hwForm.startperiod.focus()}" onMousemove="cursor('Start period can
     be first month, third quarter, etc.')" onMouseout="hidecursor()"></TD>
330  </TR>
331  <TR>
332    <TD colspan=3><br><b>Path in which the R package was installed in your
     computer:</b></TD>
333  </TR>
334  <TR>
335      <TD><input type="text" name="Rpath" value="c:/progra~1/r/rw1051"><br></TD>

336    <TD></TD>
337    <TD></TD>
338  </TR>
339  <TR>
340    <TD colspan=3><br><b>Path in which you want to save the temporary files:</b></TD>
341  </TR>
342  <TR>
343      <TD><input type="text" name="rfilepath" value="c:/temp"><br></TD>
344    <TD></TD>
345    <TD></TD>
346  </TR>
347   <TR>
348      <td colspan=3><hr></td>
349   </TR>
350  <TR>
351    <TD><b>Model type:</b></TD>
352    <TD><input type="radio" name="modelType" value="multiplicative"
     checked>Multiplicative</TD>
353    <TD><input type="radio" name="modelType" value="additive">Additive</TD>
354  </TR>
355  <TR>
356    <TD colspan=3><b>Smoothing weights: </b>(if you leave the following weights blank,
     the default weights will be chosen.)</TD>
357  </TR>
358  <TR>
359    <TD>Level(Alpha):</TD>
360    <TD>Trend(Beta):</TD>
361    <TD>Season(Gamma):</TD>
362  </TR>
363  <TR>
364    <TD><input type="text" name="levelText" onmousemove="cursor('The level weight should
     be larger than 0 and less than 1')" onmouseout="hidecursor()"></TD>
365    <TD><input type="text" name="trendText" onmousemove="cursor('The trend weight should
     be larger than 0 and less than 1')" onmouseout="hidecursor()"></TD>
366    <TD><input type="text" name="seasonText" onmousemove="cursor('The season weight
     should be larger than 0 and less than 1')" onmouseout="hidecursor()"></TD>
```

```
367   </TR>
368    <tr>
369        <td colspan=3><hr></td>
370    </tr>
371   <TR>
372     <TD colspan=3><input type="checkbox" name="checkForecast" checked
      onClick="SureForecast()"><b>Generate forecasts</b></TD>
373   </TR>
374   <TR>
375     <TD colspan=2><span id="s1" >Periods ahead of forecasting:</span>
376     </TD>
377     <TD><input type="text" name="foreNo" ID="foreNo1" value="12" ></TD>
378   </TR>
379    <tr>
380        <td colspan=3><hr></td>
381    </tr>
382   <TR>
383     <TD><b>Output:</b></TD>
384     <TD><input type="button" value="Select all"
      onClick="document.hwForm.fitdata.checked='true';document.hwForm.residata.checked='true'
      ;document.hwForm.foredata.checked='true';document.hwForm.realfit.checked='true';documen
      t.hwForm.seagraph.checked='true';document.hwForm.resigraph.checked='true';document.hwFo
      rm.realfore.checked='true';"></TD>
385       <TD><input type="button" value="Unselect all"
      onClick="document.hwForm.fitdata.checked=false;document.hwForm.residata.checked=false;d
      ocument.hwForm.foredata.checked=false;document.hwForm.realfit.checked=false;document.hw
      Form.seagraph.checked=false;document.hwForm.resigraph.checked=false;document.hwForm.rea
      lfore.checked=false;"></TD>
386   </TR>
387   <TR>
388     <TD>Data</TD>
389     <TD><input type="checkbox" name="fitdata">Fitted data</TD>
390     <TD><input type="checkbox" name="residata">Residuals</TD>
391   </TR>
392   <TR>
393     <TD></TD>
394     <TD><input type="checkbox" name="foredata" ><span id="s2">Forecast values</span></TD>

395     <TD></TD>
396   </TR>
397   <TR>
398     <TD>Graphics</TD>
399     <TD><input type="checkbox" name="realfit" checked>Plot real data & fits</TD>
400     <TD><input type="checkbox" name="seagraph">Plot seasonal index</TD>
401   </TR>
402   <TR>
403      <TD></TD>
404      <TD><input type="checkbox" name="resigraph">Plot residuals</TD>
405      <TD><input type="checkbox" name="realfore"><span id="s3">Plot real data &
      forecasts</span></TD>
406   </TR>
407    <tr>
408        <td colspan=3><hr></td>
409    </tr>
410   <tr>
411       <td></td>
412      <td><input type="button" name="runButton" value=" Go "
      onClick="checkform();doWinter()">  </td>
413         <td><input type="button" name="clearButton" value=" Clear "
      onClick="reset();document.hwForm.foreNo.value='';document.hwForm.sealength.value='';doc
      ument.hwForm.starttime.value='';document.hwForm.startperiod.value='';"></td>

414   </tr>
415
416  </TABLE>
417  </center>
418  </form>
419  </BODY>
420  </HTML>
```

Listing E.3. The tsd.html program

```
<HTML>
<HEAD>
<TITLE>Time Series Decomposition</TITLE>
<style>
TABLE {width:80%;}
span {color:black;}
</style>
<script language="Javascript">
<!--
function message()
{
alert("ATTENTION:\n"
+"The program will generate a batch file and a R command file, if you click the GO button.\n"
+"Please run the batch file to invoke R and get your text or graphical output files.\n\n"
+"You can use the example data file 'beer.dat', but make sure that you have stored it\n"
+"in the directory you want to specify to save temporary files.")
}
function DetectIE()
{
  //Detect IE version
  IEversion=0
  if (navigator.appVersion.indexOf("MSIE")!=-1)
  {
    verTemp=navigator.appVersion.split("MSIE")
    IEversion=parseFloat(verTemp[1])
  }
}
function SureForecast()
{
   //Detect IE version
   DetectIE()
 //if IE's version is 5.0+, then using special dynamic features
 if (IEversion>=5)
 {
   if (document.tsdForm.checkForecast.checked)
   {
          document.tsdForm.foreNo.disabled=false
    document.tsdForm.foredata.disabled=false
    document.tsdForm.realfore.disabled=false
    document.tsdForm.all.s1.style.color="black"
    document.tsdForm.all.s2.style.color="black"
    document.tsdForm.all.s3.style.color="black"
   }
   else
   {
    document.tsdForm.foreNo.disabled=true
    document.tsdForm.foredata.disabled=true
    document.tsdForm.realfore.disabled=true
    document.tsdForm.all.s1.style.color="gray"
    document.tsdForm.all.s2.style.color="gray"
    document.tsdForm.all.s3.style.color="gray"
   }
  }
}
function checkform()
{
 document.tsdForm.tf.value="ok"
}

function theHelp()
{
```

```
62
     helpWindow=window.open("","helpWindow","HEIGHT=300,WIDTH=300,menubar,resizeable,scrollb
     ars,status" )
63     helpWindow.document.open();
64     helpWindow.document.writeln("<html><head><title>my help ");
65     helpWindow.document.writeln("</title></head>");
66     helpWindow.document.writeln("<body><PRE>");
67     helpWindow.document.writeln("This is the help")
68     helpWindow.document.writeln("</PRE></body></html>")
69     helpWindow.document.close()
70   }
71   function cursor(text)
72   {
73      //Detect IE version
74      DetectIE()
75    //if IE's version is 5.0+, then using special dynamic features
76    if (IEversion>=5)
77    {
78        sprompt.innerHTML=text
79        sprompt.style.color="red"
80       sprompt.style.visibility="visible"
81       sprompt.style.position="absolute"
82       sprompt.style.left=event.clientX-50
83       sprompt.style.top=event.clientY+10
84    }
85   }
86
87   function hidecursor()
88   {
89      //Detect IE version
90      DetectIE()
91    //if IE's version is 5.0+, then using special dynamic features
92    if (IEversion>=5)
93    {sprompt.style.visibility="hidden"}
94   }
95   function doTSD()
96   {
97   var filename=tsdForm.filename.value
98   var sealength=tsdForm.sealength.value
99   var starttime=tsdForm.starttime.value
100  var startperiod=tsdForm.startperiod.value
101  var Rpath=tsdForm.Rpath.value
102  var rfilepath=tsdForm.rfilepath.value
103  var modeltype
104  if (tsdForm.modelType[0].checked)
105  {modeltype="multiplicative"}
106  else{modeltype="additive"}
107  if (tsdForm.modelType[1].checked)
108  {modeltype="additive"}
109  var isfore=tsdForm.checkForecast.value
110  var foreno=tsdForm.foreNo.value
111  var fitdata=tsdForm.fitdata.value
112  var residata=tsdForm.residata.value
113  var foredata=tsdForm.foredata.value
114  var realfit=tsdForm.realfit.value
115  var seagraph=tsdForm.seagraph.value
116  var resigraph=tsdForm.resigraph.value
117  var realfore=tsdForm.realfore.value
118
119  //users need to change the following variables to match their local installation

120  var Rcom=Rpath+"/bin/rterm.exe"
121  var sample=rfilepath+"/beer.dat"
122  //*********************************************************************
123  var mybat=rfilepath+"/mybat.bat"
124  var rfile=rfilepath+"/rfile.R"
125  var rlog=rfilepath+"/rlog.txt"
126  var fso1=new ActiveXObject("Scripting.FileSystemObject")
127  var myfile = fso1.CreateTextFile(mybat, true)
128  myfile.WriteLine(Rcom+" --vanilla <"+rfile+">rlog")
129  myfile.Close()
```

```
130
131  var fso2 = new ActiveXObject("Scripting.FileSystemObject");
132  var file = fso2.CreateTextFile(rfile, true);
133  file.WriteLine("#TSD")
134  file.WriteLine("library(ts)")
135  //if the user doesn't choose any file, then by default use the sample data (beer.dat)

136  if (!filename)
137  {file.WriteLine("mydata<-read.table('"+sample+"')")}
138  else
139  {file.WriteLine("mydata<-read.table('"+filename+"')")}
140  file.WriteLine("n<-nrow(mydata)")
141  file.WriteLine("c<-ncol(mydata)")
142  file.WriteLine("newc<-c+1")
143  file.WriteLine("cycle<-floor(n/"+sealength+")")
144  file.WriteLine("rest<-n-cycle*"+sealength)
145  file.WriteLine("season<-"+sealength)
146  file.WriteLine("#convert the data to time series object")
147  file.WriteLine("ts.mydata<-
     ts(mydata[,1],start=c("+starttime+","+startperiod+"),frequency="+sealength+")")

148  file.WriteLine("startTime<-"+starttime)
149  file.WriteLine("startPeriod<-"+startperiod)
150  file.WriteLine("#fit the data to TSD model")
151  file.WriteLine("mydata[,newc]<-c(1:n)")
152  file.WriteLine("lm.mydata<-lm(mydata[,1]~mydata[,newc])")
153  file.WriteLine("inte.mydata<-lm.mydata$coefficients[1]")
154  file.WriteLine("slope.mydata<-lm.mydata$coefficients[2]")
155  file.WriteLine("tsd.mydata<-decompose(ts.mydata,type='"+modeltype+"')")

156  file.WriteLine("index.mydata<-tsd.mydata$figure")
157  file.WriteLine("seasonal.mydata<-rep(index.mydata,times=cycle)")
158  file.WriteLine("doSeason<-function(rest,x,y){")
159  file.WriteLine("le<-length(x)")
160  file.WriteLine("if (rest!=0){")
161  file.WriteLine("for (i in 1:rest){")
162  file.WriteLine("x[le+i]<-y[i]")
163  file.WriteLine("x}}")
164  file.WriteLine("else {x}}")
165  file.WriteLine("seasonal2.mydata<-doSeason(rest,x=seasonal.mydata,y=index.mydata)")

166  file.WriteLine("trend.mydata<-inte.mydata+mydata[,newc]*slope.mydata")

167  //check the model type and
168  if (modeltype == "multiplicative")
169  {file.WriteLine("fit.mydata<-trend.mydata*seasonal2.mydata")}
170  else
171  {file.WriteLine("fit.mydata<-trend.mydata+seasonal2.mydata")}
172  file.WriteLine("#generate the fitted data")
173  file.WriteLine("fit.ts<-
     ts(fit.mydata,start=c(startTime,startPeriod),frequency=season)")
174  var datafile1=""
175  var datafile2=""
176  var datafile2=""
177  var datafile4=""
178  var plotfile1=""
179  var plotfile2=""
180  var plotfile3=""
181  var plotfile4=""
182  //if need to make forecast
183  if (isfore)
184  {
185   file.WriteLine("#make forecast")
186   file.WriteLine("trend.fore<-inte.mydata+c((n+1):(n+"+foreno+"))*slope.mydata")
187   file.WriteLine("doForeSea<-function(re,y,fore,sea){")
188   file.WriteLine("x<-rep(NA,times=fore)")
189   file.WriteLine("for (j in 1:fore)")
190   file.WriteLine("{if (re==0)")
191   file.WriteLine("{mm<-j%%sea}")
192   file.WriteLine("else")
193   file.WriteLine("{mm<-(j+re)%%sea}")
```

```
194   file.WriteLine("if (mm==0)")
195   file.WriteLine("{x[j]<-y[sea]}")
196   file.WriteLine("else")
197   file.WriteLine("{x[j]<-y[mm]}}")
198   file.WriteLine("x}")
199   file.WriteLine("season.fore<-
      doForeSea(re=rest,y=index.mydata,fore="+foreno+",sea=season)")
200    //check the model type
201    if (modeltype == "multiplicative")
202    {file.WriteLine("fore.mydata<-trend.fore*season.fore")}
203    else
204    {file.WriteLine("fore.mydata<-trend.fore+season.fore")}
205   file.WriteLine("doTs<-function(time,cy,re){")
206   file.WriteLine("if (re==0){")
207   file.WriteLine("mystart<-c((time+cy),1)}")
208   file.WriteLine("else{")
209   file.WriteLine("mystart<-c((time+cy-1),(re+1))}")
210   file.WriteLine("mystart}")
211   file.WriteLine("theStart<-doTs(time=startTime,cy=cycle,re=rest)")
212   file.WriteLine("fore.ts<-ts(fore.mydata,start=theStart,frequency=season)")
213   //output forecast values
214   if (foredata){
215       file.WriteLine("#output forecast values")
216     file.WriteLine("ma.tsd<-data.matrix(fore.ts)")
217     file.WriteLine("ve.tsd<-as.vector(ma.tsd)")
218
      file.WriteLine("write.table(ve.tsd,file='"+rfilepath+"/foredata.txt',quote=FALSE,row.na
      mes=FALSE,col.names=c('The forecast values:'))")
219      var datafile1=rfilepath+"/foredata.txt"
220   }
221   //plot real data and forecasts
222    if (realfore){
223      file.WriteLine("#plotting the real data and forecasting values")
224      file.WriteLine("bmp(file='"+rfilepath+"/realfore.bmp',width=300,height=300)")
225      file.WriteLine("ts.plot(ts.mydata,fit.ts,fore.ts,col=c('blue','green','red'))")
226      file.WriteLine("title(main='The plot of real data and forecasts (real data: blue,
      fits: green, forecasts: red)')")
227      file.WriteLine("dev.off()")
228     var plotfile1=rfilepath+"/realfore.bmp"
229    }
230   }
231   //output fitted data
232    if (fitdata){
233      file.WriteLine("#output fitted data")
234      file.WriteLine("ma.fit<-data.matrix(fit.ts)")
235      file.WriteLine("ve.fit<-as.vector(ma.fit)")
236
      file.WriteLine("write.table(ve.fit,file='"+rfilepath+"/fitdata.txt',quote=FALSE,row.nam
      es=FALSE,col.names=c('The fitted data:'))")
237     var datafile2=rfilepath+"/fitdata.txt"
238   }
239   file.WriteLine("re.mydata<-ts.mydata-fit.ts")
240   //output residuals
241   if (residata){
242     file.WriteLine("#output residuals")
243     file.WriteLine("ma.re<-data.matrix(re.mydata)")
244     file.WriteLine("ve.re<-as.vector(ma.re)")
245
      file.WriteLine("write.table(ve.re,file='"+rfilepath+"/residata.txt',quote=FALSE,row.nam
      es=FALSE,col.names=c('The residuals:'))")
246    var datafile3=rfilepath+"/residata.txt"
247   }
248
249   //plot real data and fits
250   if (realfit){
251     file.WriteLine("#plotting the real data and fits")
252     file.WriteLine("bmp(file='"+rfilepath+"/realfit.bmp',width=300,height=300)")
253     file.WriteLine("ts.plot(ts.mydata,fit.ts,col=c('blue','red'))")
254     file.WriteLine("title(main='The plot of real data and fits (real data: blue, fits:
      red)')")
255     file.WriteLine("dev.off()")
```

```
 var plotfile2=rfilepath+"/realfit.bmp"
}
//plot residuals
if (resigraph){
  file.WriteLine("#plotting the residuals")
  file.WriteLine("bmp(file='"+rfilepath+"/resigraph.bmp',width=300,height=300)")
  file.WriteLine("plot(re.mydata,col=c('blue'))")
  file.WriteLine("title(main='The plot of residuals')")
  file.WriteLine("dev.off()")
 var plotfile3=rfilepath+"/resigraph.bmp"
}
//plot seasonal index
if (seagraph){
  file.WriteLine("#plotting the seasonal index")
  file.WriteLine("index<-tsd.mydata$figure")
  file.WriteLine("bmp(file='"+rfilepath+"/seagraph.bmp',width=300,height=300)")
  file.WriteLine("plot(index, type='b')")
  file.WriteLine("title(main='The plot of seasonal indices')")
  file.WriteLine("dev.off()")
   var plotfile4=rfilepath+"/seagraph.bmp"
}
file.WriteLine("myresult<-
matrix(c(tsd.mydata$type,rep('',11),tsd.mydata$figure[1:12],inte.mydata,rep('',11),slop
e=c(slope.mydata),rep('',11)),nrow=4,ncol=12,byrow=TRUE)")

file.WriteLine("write.table(myresult,file='"+rfilepath+"/myresult.txt',quote=FALSE,row.
names=c('model type:','seasonal indices:','intercept of the treand equation:','slope of
the trend equation:'),col.names=c('The final results:',rep('',11)))n")

var datafile4=rfilepath+"/myresult.txt"
file.WriteLine("")
file.Close()
var datafile=datafile1+" "+datafile2+" "+datafile3+" "+datafile4
var plotfile=plotfile1+" "+plotfile2+" "+plotfile3+" "+plotfile
alert("Now please run the batch file '"+mybat+"' to invoke R! \n\n"
+"Then you will get the following files in c:/temp: \n"
+"1. The file which performs forecasting and plotting: "+rfile+" ;\n"
+"2. The R log file:"+rlog+" ;\n"
+"3. The plots include: "+plotfile+" ;\n"
+"4. The output data files include: "+datafile+" .\n\n"
+"ATTENTION: please move or rename the above files for the future use, otherwise next
operation will overwrite them!!!")
}
//-->
</script>
</HEAD>
<BODY onload="message()">
<form name="tsdForm" action="tsd.cgi" enctype="multipart/form-data" method=post>
<center><h3>Time Series Decomposition</h3></center>
<center>
<TABLE>
  <TR>
   <td></td>
   <td align=center><input type="button" value="Help" onclick="theHelp()"></td>
   <td align=right><A href="localfore.html">Home </A></td>
</TR>
  <TR>
     <td colspan=3><hr></td>
  </TR>
<TR>
  <TD colspan=3><b>Load data: </b>(if you don't choose a local file, the sample data
will be used.)</TD>
</TR>
<TR>
  <TD>Choose a local file: </TD>
  <TD colspan=2><INPUT type="file" name="filename"></TD>
</TR>
<TR>
  <TD colspan=3><b><BR>Transfer the data to a time series:</b></TD>
</TR>
<TR><span id="sprompt" style="visibility:hidden">my prompts</span></TR>
```

```
<TR>
  <TD>Length of season: </TD>
  <TD>Start time: </TD>
  <TD>Start period: </TD>
</TR>
<TR>
  <TD><INPUT type="text" name="sealength" value="12" onBlur="if
(!document.tsdForm.sealength.value){alert('A season length is
required!');document.tsdForm.sealength.focus()}"  onMousemove="cursor('Length of season
can be 12 months per year, 4 quarters per year, etc.')" onMouseout="hidecursor()"></TD>

  <TD><INPUT type="text" name="starttime" value="1956" onBlur="if
(!document.tsdForm.starttime.value){alert('A start time is
required!');document.tsdForm.starttime.focus()}" onMousemove="cursor('Start time can be
year 1956, etc.')" onMouseout="hidecursor()"></TD>
    <TD><INPUT type="text" name="startperiod" value="1" onBlur="if
(!document.tsdForm.startperiod.value){alert('A start period is
required!');document.tsdForm.startperiod.focus()}" onMousemove="cursor('Start period
can be first month, third quarter, etc.')" onMouseout="hidecursor()"></TD>
</TR>
<TR>
  <TD colspan=3><br><b>Path in which the R package was installed in your
computer:</b></TD>
</TR>
<TR>
    <TD><input type="text" name="Rpath" value="c:/progra~1/r/rw1051"><br></TD>
  <TD></TD>
  <TD></TD>
</TR>
<TR>
  <TD colspan=3><br><b>Path in which you want to save the temporary files:</b></TD>
</TR>
<TR>
    <TD><input type="text" name="rfilepath" value="c:/temp"><br></TD>
  <TD></TD>
  <TD></TD>
</TR>
 <TR>
    <td colspan=3><hr></td>
 </TR>
<TR>
  <TD><b>Model type:</b></TD>
  <TD><input type="radio" name="modelType" value="multiplicative"
checked>Multiplicative</TD>
  <TD><input type="radio" name="modelType" value="additive">Additive</TD>
</TR>
 <tr>
     <td colspan=3><hr></td>
 </tr>
<TR>
  <TD colspan=3><input type="checkbox" name="checkForecast" checked
onClick="SureForecast()"><b>Generate forecasts</b></TD>
</TR>
<TR>
  <TD colspan=2><span id="s1" >Periods ahead of forecasting:</span>
  </TD>
  <TD><input type="text" name="foreNo" ID="foreNo1" value="12" ></TD>
</TR>
 <tr>
    <td colspan=3><hr></td>
 </tr>
<TR>
  <TD><b>Output:</b></TD>
  <TD><input type="button" value="Select all"
onClick="document.tsdForm.fitdata.checked='true';document.tsdForm.residata.checked='tru
e';document.tsdForm.foredata.checked='true';document.tsdForm.realfit.checked='true';doc
ument.tsdForm.seagraph.checked='true';document.tsdForm.resigraph.checked='true';documen
t.tsdForm.realfore.checked='true';"></TD>
    <TD><input type="button" value="Unselect all"
onClick="document.tsdForm.fitdata.checked=false;document.tsdForm.residata.checked=false
;document.tsdForm.foredata.checked=false;document.tsdForm.realfit.checked=false;documen
```

```
t.tsdForm.seagraph.checked=false;document.tsdForm.resigraph.checked=false;document.tsdF
orm.realfore.checked=false;"></TD>
371  </TR>
372  <TR>
373    <TD>Data</TD>
374    <TD><input type="checkbox" name="fitdata">Fitted data</TD>
375    <TD><input type="checkbox" name="residata">Residuals</TD>
376  </TR>
377  <TR>
378    <TD></TD>
379    <TD><input type="checkbox" name="foredata" ><span id="s2">Forecast values</span></TD>
380    <TD></TD>
381  </TR>
382  <TR>
383    <TD>Graphics</TD>
384    <TD><input type="checkbox" name="realfit" checked>Plot real data & fits</TD>
385    <TD><input type="checkbox" name="seagraph">Plot seasonal index</TD>
386  </TR>
387  <TR>
388     <TD></TD>
389     <TD><input type="checkbox" name="resigraph">Plot residuals</TD>
390     <TD><input type="checkbox" name="realfore"><span id="s3">Plot real data &
    forecasts</span></TD>
391  </TR>
392   <tr>
393       <td colspan=3><hr></td>
394   </tr>
395  <tr>
396       <td><input type="hidden" name="tf" value=""></td>
397     <td><input type="button" name="runButton" value=" Go "
    onClick="checkform();doTSD()"">  </td>
398         <td><input type="button" name="clearButton" value=" Clear "
    onClick="reset();document.tsdForm.foreNo.value='';document.tsdForm.sealength.value='';d
    ocument.tsdForm.starttime.value='';document.tsdForm.startperiod.value='';"></td>

399  </tr>
400 </TABLE>
401 </center>
402 </form>
403 </BODY>
404 </HTML>
```

Listing E.4. The arima.html program

```
<HTML>
<HEAD>
<TITLE>ARIMA</TITLE>
<style>
TABLE {width:80%;}
span {color:black;}
</style>
<script language="Javascript">
<!--
function message()
{
alert("ATTENTION:\n"
+"The program will generate a batch file and a R command file, if you click the GO button.\n"
+"Please run the batch file to invoke R and get your text or graphical output files.\n\n"
+"You can use the example data file 'beer.dat', but make sure that you have stored it\n"
+"in the directory you want to specify to save temporary files.")
}
function DetectIE()
{
  //Detect IE version
  IEversion=0
  if (navigator.appVersion.indexOf("MSIE")!=-1)
  {
    verTemp=navigator.appVersion.split("MSIE")
    IEversion=parseFloat(verTemp[1])
  }
}
function sureSeason()
{
   //Detect IE version
   DetectIE()
 //if IE's version is 5.0+, then using special dynamic features
 if (IEversion>=5)
 {
    if (arimaForm.IsSeason.checked)
   {
    arimaForm.ARP.disabled=false
    arimaForm.DiffD.disabled=false
    arimaForm.MAQ.disabled=false
    arimaForm.all.s4.style.color="black"
    arimaForm.all.s6.style.color="black"
   }
   else
   {
    arimaForm.ARP.value="0"
    arimaForm.ARP.disabled=true
    arimaForm.DiffD.value="0"
    arimaForm.DiffD.disabled=true
    arimaForm.MAQ.value="0"
    arimaForm.MAQ.disabled=true
    arimaForm.all.s4.style.color="gray"
    arimaForm.all.s6.style.color="gray"
   }
 }
}
function SureForecast()
{
   //Detect IE version
   DetectIE()
 //if IE's version is 5.0+, then using special dynamic features
 if (IEversion>=5)
 {
   if (document.arimaForm.checkForecast.checked)
```

```
64      {
65       document.arimaForm.foreNo.disabled=false
66       document.arimaForm.foredata.disabled=false
67       document.arimaForm.realfore.disabled=false
68       document.arimaForm.all.s1.style.color="black"
69       document.arimaForm.all.s2.style.color="black"
70       document.arimaForm.all.s3.style.color="black"
71      }
72      else
73      {
74       document.arimaForm.foreNo.disabled=true
75       document.arimaForm.foredata.disabled=true
76       document.arimaForm.realfore.disabled=true
77       document.arimaForm.all.s1.style.color="gray"
78       document.arimaForm.all.s2.style.color="gray"
79       document.arimaForm.all.s3.style.color="gray"
80      }
81    }
82   }
83   function checkform()
84   {
85    var ARp=document.arimaForm.ARp.value
86    var ARP=document.arimaForm.ARP.value
87    var Diffd=document.arimaForm.Diffd.value
88    var DiffD=document.arimaForm.DiffD.value
89    var MAq=document.arimaForm.MAq.value
90    var MAQ=document.arimaForm.MAQ.value
91    var p=1
92    if (ARp==""){p=0}else{if (ARp<0){p=0}else{ if (ARp>5){p=0}else{p=1}}}
93    var P=1
94    if (ARP==""){P=0}else{if (ARP<0){P=0}else{ if (ARP>5){P=0}else{P=1}}}
95    var d=1
96    if (Diffd==""){d=0}else{if (Diffd<0){d=0}else{ if (Diffd>5){d=0}else{d=1}}}
97    var D=1
98    if (DiffD==""){D=0}else{if (DiffD<0){D=0}else{ if (DiffD>5){D=0}else{D=1}}}
99    var q=1
100   if (MAq==""){q=0}else{if (MAq<0){q=0}else{ if (MAq>5){q=0}else{q=1}}}
101   var Q=1
102   if (MAQ==""){Q=0}else{if (MAQ<0){Q=0}else{ if (MAQ>5){Q=0}else{Q=1}}}
103   if (p==0){alert("Invalid parameters!  AR, Difference and MA should be larger than 0 and
      less than 5.")}
104   else
105   {
106     if (P==0){alert("Invalid parameters!  AR, Difference and MA should be larger than 0
      and less than 5.")}
107     else
108     {
109       if (d==0){alert("Invalid parameters!  AR, Difference and MA should be larger than 0
      and less than 5.")}
110       else
111     {
112       if (D==0){alert("Invalid parameters!  AR, Difference and MA should be larger than 0
      and less than 5.")}
113          else
114       {
115         if (q==0){alert("Invalid parameters!  AR, Difference and MA should be larger than
      0 and less than 5.")}
116            else
117            {
118              if (Q==0){alert("Invalid parameters!  AR, Difference and MA should be
      larger than 0 and less than 5.")}
119              else
120              {document.arimaForm.tf.value="ok"}
121            }
122         }
123       }
124     }
125   }
126  }
127  function theHelp()
128  {
```

```
129
    helpWindow=window.open("","helpWindow","HEIGHT=300,WIDTH=300,menubar,resizeable,scrollb
    ars,status" )
130   helpWindow.document.open();
131   helpWindow.document.writeln("<html><head><title>my help ");
132   helpWindow.document.writeln("</title></head>");
133   helpWindow.document.writeln("<body><PRE>");
134   helpWindow.document.writeln("This is the help")
135   helpWindow.document.writeln("</PRE></body></html>")
136   helpWindow.document.close()
137 }
138 function cursor(text)
139 {
140    //Detect IE version
141    DetectIE()
142  //if IE's version is 5.0+, then using special dynamic features
143  if (IEversion>=5)
144  {
145     sprompt.innerHTML=text
146     sprompt.style.color="red"
147     sprompt.style.visibility="visible"
148     sprompt.style.position="absolute"
149     sprompt.style.left=event.clientX-50
150     sprompt.style.top=event.clientY+5
151  }
152 }
153
154 function hidecursor()
155 {
156    //Detect IE version
157    DetectIE()
158  //if IE's version is 5.0+, then using special dynamic features
159  if (IEversion>=5)
160  {sprompt.style.visibility="hidden"}
161 }
162 function doARIMA()
163 {
164 var filename=arimaForm.filename.value
165 var sealength=arimaForm.sealength.value
166 var starttime=arimaForm.starttime.value
167 var startperiod=arimaForm.startperiod.value
168 var Rpath=arimaForm.Rpath.value
169 var rfilepath=arimaForm.rfilepath.value
170 var isseason=arimaForm.IsSeason.value
171 var ARp=arimaForm.ARp.value
172 var ARP=arimaForm.ARP.value
173 var Diffd=arimaForm.Diffd.value
174 var DiffD=arimaForm.DiffD.value
175 var MAq=arimaForm.MAq.value
176 var MAQ=arimaForm.MAQ.value
177 var isconstant=arimaForm.Isconstant.value
178 var isfore=arimaForm.checkForecast.value
179 var foreno=arimaForm.foreNo.value
180 var residata=arimaForm.residata.value
181 var foredata=arimaForm.foredata.value
182 var diaggraph=arimaForm.diaggraph.value
183 var realfore=arimaForm.realfore.value
184 //users need to change the following variables to match their local installation

185 var Rcom=Rpath+"/bin/rterm.exe"
186 var sample=rfilepath+"/beer.dat"
187 //*********************************************************************

188 var mybat=rfilepath+"/mybat.bat"
189 var rfile=rfilepath+"/rfile.R"
190 var rlog=rfilepath+"/rlog.txt"
191 var fso1=new ActiveXObject("Scripting.FileSystemObject")
192 var myfile = fso1.CreateTextFile(mybat, true)
193 myfile.WriteLine(Rcom+" --vanilla <"+rfile+">rlog")
194 myfile.Close()
195 var fso2 = new ActiveXObject("Scripting.FileSystemObject");
```

```
var file = fso2.CreateTextFile(rfile, true);
file.WriteLine("#ARIMA")
file.WriteLine("library(ts)")
//if the user doesn't choose any file, then by default use the sample data (beer.dat)

if (!filename)
{file.WriteLine("mydata<-read.table('"+sample+"')")}
else
{file.WriteLine("mydata<-read.table('"+filename+"')")}
file.WriteLine("n<-nrow(mydata)")
file.WriteLine("c<-ncol(mydata)")
file.WriteLine("newc<-c+1")
file.WriteLine("cycle<-floor(n/"+sealength+")")
file.WriteLine("rest<-n-cycle*"+sealength)
file.WriteLine("season<-"+sealength)
file.WriteLine("#convert the data to time series object")
file.WriteLine("ts.mydata<-
ts(mydata[,1],start=c("+starttime+","+startperiod+"),frequency="+sealength+")")

file.WriteLine("startTime<-"+starttime)
file.WriteLine("startPeriod<-"+startperiod)
file.WriteLine("#fit the data to ARIMA model")
if (isconstant)
{mymean="TRUE";}
else
{mymean="FALSE";}
if (isseason)
{file.WriteLine("arima.mydata<-
arima(ts.mydata,order=c("+ARp+","+Diffd+","+MAq+"),seasonal=list(order=c("+ARP+","+Diff
D+","+MAQ+")),include.mean="+mymean+")")}
else
{file.WriteLine("arima.mydata<-
arima(ts.mydata,order=c("+ARp,"+Diffd,"+MAq+"),include.mean="+mymean+")")}

var datafile1=""
var datafile2=""
var datafile3=""
var plotfile1=""
var plotfile2=""
//if need to make forecast
if (isfore)
{
    file.WriteLine("#make forecast")
    file.WriteLine("predict.mydata<-predict(arima.mydata, n.ahead="+foreno+")")
    file.WriteLine("fore.mydata<-predict.mydata$pred")
 //output forecast values
 if (foredata){
     file.WriteLine("#output forecast values")
     file.WriteLine("ma.fore<-data.matrix(fore.mydata)")
    file.WriteLine("ve.fore<-as.vector(ma.fore)")

file.WriteLine("write.table(ve.fore,file='"+rfilepath+"/foredata.txt',quote=FALSE,row.n
ames=FALSE,col.names=c('The forecast values:'))")
    var datafile1=rfilepath+"/foredata.txt"
   }
 //plot real data and forecasts
  if (realfore){
    file.WriteLine("#plotting the real data and forecasting values")
    file.WriteLine("bmp(file='"+rfilepath+"/realfore.bmp',width=300,height=300)")
    file.WriteLine("ts.plot(ts.mydata,fore.mydata,col=c('blue','red'))")
    file.WriteLine("title(main='The plot of real data and forecasts (real data: blue,
fits: green, forecasts: red)')")
    file.WriteLine("dev.off()")
   var plotfile1=rfilepath+"/realfore.bmp"
  }
}
file.WriteLine("resi.mydata<-arima.mydata$residuals")
//output residuals
if (residata){
  file.WriteLine("#output residuals")
  file.WriteLine("ma.resi<-data.matrix(resi.mydata)")
```

```
257    file.WriteLine("ve.resi<-as.vector(ma.resi)")
258
       file.WriteLine("write.table(ve.resi,file='"+rfilepath+"/residata.txt',quote=FALSE,row.n
       ames=FALSE,col.names=c('The residuals:'))")
259    var datafile2=rfilepath+"/residata.txt"
260    }
261    //#lot tsdiag
262    if (diaggraph){
263      file.WriteLine("#plotting tsdiag")
264      file.WriteLine("bmp(file='"+rfilepath+"/diaggraph.bmp',width=300,height=500)")

265      file.WriteLine("tsdiag(arima.mydata)")
266      file.WriteLine("dev.off()")
267    var plotfile1=rfilepath+"/diaggraph.bmp"
268    }
269    file.WriteLine("coef<-length(arima.mydata$coef)")
270    file.WriteLine("thel<-7")
271    file.WriteLine("if (coef>8){thel<-(coef-1)}")
272    file.WriteLine("myresult<-matrix(c(arima.mydata$aic,rep('',(thel-
       1)),arima.mydata$arma[1:7],rep('',(thel-7)),names(arima.mydata$coef)[1:(coef-
       1)],rep('',(8-coef)),arima.mydata\$coef[1:(coef-1)],rep(\"\",(8-
       coef))),nrow=4,ncol=thel,byrow=TRUE)")
273
       file.WriteLine("write.table(myresult,file='"+rfilepath+"/myresult.txt',quote=FALSE,row.
       names=c('AIC:','arima[p,q,P,Q,L,d,D]:','names of coefficients:','values of
       coefficients:'),col.names=c('The final results:',rep('',6)))")
274    var datafile3=rfilepath+"/myresult.txt"
275    file.WriteLine("")
276    file.Close()
277    var datafile=datafile1+" "+datafile2+" "+datafile3
278    var plotfile=plotfile1+" "+plotfile2
279    alert("Now please run the batch file '"+mybat+"' to invoke R! \n\n"
280    +"Then you will get the following files in c:/temp: \n"
281    +"1. The file which performs forecasting and plotting: "+rfile+" ;\n"
282    +"2. The R log file:"+rlog+" ;\n"
283    +"3. The plots include: "+plotfile+" ;\n"
284    +"4. The output data files include: "+datafile+" .\n\n"
285    +"ATTENTION: please move or rename the above files for the future use, otherwise next
       operation will overwrite them!!!")
286  }
287  //-->
288  </script>
289  </HEAD>
290  <BODY onload="message()">
291  <form name="arimaForm" action="arima.cgi" enctype="multipart/form-data" method=post>

292  <center><h3>ARIMA modeling</h3></center>
293  <center>
294  <TABLE>
295    <TR>
296     <td></td>
297     <td align=center><input type="button" value="Help" onclick="theHelp()"></td>

298     <td align=right><A href="localfore.html">Home </A></td>
299   </TR>
300    <TR>
301       <td colspan=3><hr></td>
302    </TR>
303   <TR>
304     <TD colspan=3><b>Load data: </b>(if you don't choose a local file, the sample data
       will be used.)</TD>
305   </TR>
306   <TR>
307     <TD>Choose a local file: </TD>
308     <TD colspan=2><INPUT type="file" name="filename"></TD>
309   </TR>
310   <TR>
311     <TD colspan=3><b><BR>Transfer the data to a time series:</b></TD>
312   </TR>
313   <TR><span id="sprompt" style="visibility:hidden">my prompts</span></TR>
314   <TR>
```

```
315     <TD>Length of season: </TD>
316     <TD>Start time: </TD>
317     <TD>Start period: </TD>
318   </TR>
319   <TR>
320     <TD><INPUT type="text" name="sealength" value="12" onBlur="if
      (!document.arimaForm.sealength.value){alert('A season length is
      required!');document.arimaForm.sealength.focus()}"  onMousemove="cursor('Length of
      season can be 12 months per year, 4 quarters per year, etc.')"
      onMouseout="hidecursor()"></TD>
321     <TD><INPUT type="text" name="starttime" value="1956" onBlur="if
      (!document.arimaForm.starttime.value){alert('A start time is
      required!');document.arimaForm.starttime.focus()}" onMousemove="cursor('Start time can
      be year 1956, etc.')" onMouseout="hidecursor()"></TD>
322       <TD><INPUT type="text" name="startperiod" value="1" onBlur="if
      (!document.arimaForm.startperiod.value){alert('A start period is
      required!');document.arimaForm.startperiod.focus()}" onMousemove="cursor('Start period
      can be first month, third quarter, etc.')" onMouseout="hidecursor()"></TD>
323   </TR>
324   <TR>
325     <TD colspan=3><br><b>Path in which the R package was installed in your
      computer:</b></TD>
326   </TR>
327   <TR>
328       <TD><input type="text" name="Rpath" value="c:/progra~1/r/rw1051"><br></TD>

329     <TD></TD>
330     <TD></TD>
331   </TR>
332   <TR>
333     <TD colspan=3><br><b>Path in which you want to save the temporary files:</b></TD>
334   </TR>
335   <TR>
336       <TD><input type="text" name="rfilepath" value="c:/temp"><br></TD>
337     <TD></TD>
338     <TD></TD>
339   </TR>
340    <TR>
341       <td colspan=3><hr></td>
342    </TR>
343    <tr>
344      <td></td>
345      <td></td>
346      <td><input type="checkbox" name="IsSeason" checked onClick="sureSeason()"><span
      id="s4" >Fit seasonal model</span></td>
347   </tr>
348    <tr>
349      <td></td>
350      <td>Nonseasonal</td>
351      <td><span id="s6" >Seasonal</span></td>
352   </tr>
353   <TR>
354     <TD>Autoregessive(AR)</TD>
355     <TD><input type="text" name="ARp" onMousemove="cursor('AR should be larger than 0 and
      less than 5.')" onMouseout="hidecursor()"></TD>
356     <TD><input type="text" name="ARP" onMousemove="cursor('AR should be larger than 0 and
      less than 5.')" onMouseout="hidecursor()"></TD>
357   </TR>
358   <TR>
359     <TD>Difference</TD>
360     <TD><input type="text" name="Diffd" onMousemove="cursor('Differenc should be larger
      than 0 and less than 5.')" onMouseout="hidecursor()"></TD>
361     <TD><input type="text" name="DiffD" onMousemove="cursor('Differenc should be larger
      than 0 and less than 5.')" onMouseout="hidecursor()"></TD>
362   </TR>
363   <TR>
364     <TD>Moving average(MA)</TD>
365     <TD><input type="text" name="MAq" onMousemove="cursor('MA should be larger than 0 and
      less than 5.')" onMouseout="hidecursor()"></TD>
366     <TD><input type="text" name="MAQ" onMousemove="cursor('MA should be larger than 0 and
      less than 5.')" onMouseout="hidecursor()"></TD>
```

```
</TR>
  <tr>
     <td colspan=3><input type="checkbox" name="Isconstant" checked>Include constant term in model</td>
  </tr>
  <tr>
     <td colspan=3><hr></td>
  </tr>
<TR>
  <TD colspan=3><input type="checkbox" name="checkForecast" checked onClick="SureForecast()"><b>Generate forecasts</b></TD>
</TR>
<TR>
  <TD colspan=2><span id="s1" >Periods ahead of forecasting:</span>
  </TD>
  <TD><input type="text" name="foreNo" ID="foreNo1" value="12" ></TD>
</TR>
  <tr>
     <td colspan=3><hr></td>
  </tr>
<TR>
  <TD><b>Output:</b></TD>
  <TD><input type="button" value="Select all" onClick="document.arimaForm.residata.checked='true';document.arimaForm.foredata.checked='true';document.arimaForm.diaggraph.checked='true';document.arimaForm.realfore.checked='true';"></TD>
     <TD><input type="button" value="Unselect all" onClick="document.arimaForm.residata.checked=false;document.arimaForm.foredata.checked=false;document.arimaForm.diaggraph.checked=false;document.arimaForm.realfore.checked=false;"></TD>
</TR>
<TR>
  <TD>Data</TD>
  <TD><input type="checkbox" name="residata">Residuals</TD>
  <TD><input type="checkbox" name="foredata" ><span id="s2">Forecast values</span></TD>
</TR>
<TR>
  <TD>Graphics</TD>
  <TD><input type="checkbox" name="diaggraph" checked>Plot diag graph</TD>
     <TD><input type="checkbox" name="realfore" ><span id="s3">Plot real data & forecasts</span></TD>
</TR>
  <tr>
     <td colspan=3><hr></td>
  </tr>
<tr>
     <td><input type="hidden" name="tf" value=""></td>
   <td><input type="button" name="runButton" value=" Go " onClick="checkform();doARIMA()">  </td>
      <td><input type="button" name="clearButton" value=" Clear " onClick="reset();document.arimaForm.foreNo.value='';document.arimaForm.sealength.value='';document.arimaForm.starttime.value='';document.arimaForm.startperiod.value='';"></td>
</tr>
</TABLE>
</center>
</form>
</BODY>
</HTML>
```