Studies in Fuzziness and Soft Computing

Farhad Hosseinzadeh Lotfi
Ali Ebrahimnejad
Mohsen Vaez-Ghasemi
Zohreh Moghaddas

Data Envelopment Analysis with R

Springer

Studies in Fuzziness and Soft Computing

Volume 386

Series Editor

Janusz Kacprzyk, Polish Academy of Sciences, Systems Research Institute, Warsaw, Poland

The series "Studies in Fuzziness and Soft Computing" contains publications on various topics in the area of soft computing, which include fuzzy sets, rough sets, neural networks, evolutionary computation, probabilistic and evidential reasoning, multi-valued logic, and related fields. The publications within "Studies in Fuzziness and Soft Computing" are primarily monographs and edited volumes. They cover significant recent developments in the field, both of a foundational and applicable character. An important feature of the series is its short publication time and world-wide distribution. This permits a rapid and broad dissemination of research results. Contact the series editor by e-mail: kacprzyk@ibspan.waw.pl

Indexed by ISI, DBLP and Ulrichs, SCOPUS, Zentralblatt Math, GeoRef, Current Mathematical Publications, IngentaConnect, MetaPress and Springerlink.
The books of the series are submitted for indexing to Web of Science.

More information about this series at http://www.springer.com/series/2941

Farhad Hosseinzadeh Lotfi ·
Ali Ebrahimnejad · Mohsen Vaez-Ghasemi ·
Zohreh Moghaddas

Data Envelopment Analysis with R

Springer

Farhad Hosseinzadeh Lotfi
Department of Mathematics, Science and Research Branch
Islamic Azad University
Tehran, Iran

Ali Ebrahimnejad
Department of Mathematics, Qaemshahr Branch
Islamic Azad University
Qaemshahr, Iran

Mohsen Vaez-Ghasemi
Department of Mathematics, Rasht Branch
Islamic Azad University
Rasht, Iran

Zohreh Moghaddas
Department of Mathematics, Qazvin Branch
Islamic Azad University
Qazvin, Iran

ISSN 1434-9922 ISSN 1860-0808 (electronic)
Studies in Fuzziness and Soft Computing
ISBN 978-3-030-24276-3 ISBN 978-3-030-24277-0 (eBook)
https://doi.org/10.1007/978-3-030-24277-0

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

In memory of our teacher,
Prof. Gholamreza Jahanshaloo
Frahad dedicated this book to his parents
Ali dedicated this book to his parents,
his wife (Zeynab) and his son (Arvin)
Mohsen and Zoherh dedicated this book
to their son, Bardia

Preface

The data envelopment analysis (DEA) technique is used to identify the strengths and weaknesses of a set of decision-maker units. In this technique for each decision-making unit based on the data analysis, relative efficiency, pattern coordinates, rankings, congestion in inputs, return on the scale, Malmquist productivity index, cost efficiency, and profit efficiency can be evaluated. For each decision-making unit (DMU) belonging to a set of observed units, it is necessary to solve an independent optimization problem in order to achieve the above objectives.

Suppose a n-member set of decision making units is evaluated. In this case,

- In order to assess the efficiency of all units, at least n independent optimization problems should be solved.
- For the ranking all units, at least n independent optimization problems should be solved.
- To calculate the input congestion for all units, at least $2n$ independent optimization problems should be solved.
- To calculate the return to the scale for all units, at least $2n$ independent optimization problems should be solved.
- To calculate the Malmquist productivity index for each of the n units, at least $4n$ independent optimization problems should be solved.
- To calculate the cost efficiency for all units, at least $2n$ independent optimization problems should be solved.
- To calculate the revenue and profit efficiencies for all units, at least n independent optimization problems should be solved.

In many real applications like evaluating companies registered in a country or a geographic area and evaluating faculties of a country, it has been observed that the number of DMUs can reach thousands of units. Problems designed to achieve these goals include linear programming, nonlinear programming, and mixed planning. In such problems, the number of constraints or variables is equal to the number of DMUs under evaluation where if the structure of the problem changes from a box to network, the number of constraints will multiply. This information indicates in order to achieve the objectives of the performance evaluation for a set of DMUS, a

huge number of optimization problem should be solved and this issue can be done by the help of strong software. It should be noted that when the number of DMUs as well as assessment periods is increased, the volume of results obtained from the optimization model will also increase. For example, if 2000 DMUs are evaluated within 12 months, then 24,000 efficiency scores will obtain after solving the corresponding models. In this way, it is necessary to provide an appropriate analysis for the management by analyzing the results. Therefore, all analysts working in the field of DEA need to know a software that can solve many of the optimization problems (linear, nonlinear, and so on) and have high capability for statistical analysis of the results derived of optimization models. One of the best software that can have good performance in solving optimization models and statistical analysis of the obtained results is R software.

Mathematical science is one of the most accurate and reliable sciences, whose precision of calculation from the origin of this science up to now has never been overlooked by anyone. But gradually due to the complexity of calculations and solving equations, manual calculations were not possible in terms of accuracy and time. However with the advent of computers, the speed of problem solving has grown steadily. Note that, the specialized problems were not solvable due to their very high time consuming and so the professional software came in. With the advent of such software, the knot of many unfinished and timely issues was opened, but learning such software turned into a problem for researchers. Today, the use of specialized and advanced software is one of the requirements of the work of the researchers in achieving the goals of the research and finding correct, quick, and accurate conclusions.

R is a mathematical and object-oriented programming language designed primarily for statistical computing and data mining. This language is in fact the S version of the open-source version. The R project was conducted by Robert Gentleman and Ross Ihaka in 1995 at the Auckland University's Department of Statistics, hence the name R. This application is licensed under the GNU General Public License (GNU GPL) of the Free Software Foundation and is available free of charge. It is currently being developed by a group of statisticians called the "Software Core Team" R page at www.r-project.org. Although R is often used for statistical computations, it can also be used for matrix computations, which is similar to software such as Octave and MATLAB. The programming language R covers a wide range of optimization of linear and nonlinear models, integer and quadratic programming as well as classical statistical tests, time series analysis, classification and clustering, and has high graphical capabilities. In the R environment, C, C ++, and FORTRAN have the ability to connect and invoke the program while the expert users can directly change the R objects by using the C code. Among the features of R can be graphical facilities for data analysis and drawing diagrams, a simple and advanced programming language including conditional expressions, loops, recursive functions, and a strong set of computing operators, arrays and matrices, software packages for data mining and analysis, storage, retrieval and manipulation of data, a multi-purpose library for analytical

operations, and data mining libraries. This software is free and open source. In this way, there is a special interest to this software among scholars.

DEA with precise and imprecise data has many applications in various fields such as agriculture, environment, companies, banks, insurance, universities, schools, manpower, hospitals, countries, mechanical engineering, service, and manufacturing centers. Thus, this technique can be used for most academic disciplines such as mathematics, industrial engineering, management, accounting, finance, agriculture, the environment as a tool for analyzing results. Consequently, this book is dedicated to the use of R codes for DEA models with crisp and fuzzy data. By describing crisp and fuzzy DEA models, it represents an outstanding reference guide helping experts to solve these models with R they must face in their different work environments.

The book is divided into five chapters. In the first, a brief introduction is given of the key concepts of DEA and fuzzy sets that are most relevant for the models that are considered in the rest of the volume. The following chapter is presented to some basic definitions about R and required commands used in writing DEA models with R. Following that, the third chapter covers basic models of DEA technique as well as R codes of the models along with numerical examples. In addition to the classic performance evaluation models in DEA, developed models have been introduced in this technique that help different aspects of analytics in performance evaluation. Therefore, in the fourth chapter, the developed DEA models are introduced briefly and then the corresponding R code for these models are provided. Finally, one of the models that is most frequently the motive for analytics is considered: the fuzzy DEA (FDEA) models. The conventional DEA models require precise input and output data, which may not always be available in real-world applications. However, in real-life problems, inputs and outputs are often imprecise which can be formulated with fuzzy data. Consequently, in the fifth chapter, the main approaches for solving FDEA models are classified and the mathematical approaches of each category are described briefly. Then, R codes for each FDEA model are provided.

To conclude, we the authors wish to express our special recognition to Prof. Janusz Kacprzyk, and Dr. Leontina Di Cecco who from the very first moment accepted our book proposal and who encouraged us continually throughout the preparation of this volume. And needless to say, the authors wish to manifest our sincere gratitude to our respective families for the support, understanding, and patience that they have shown us throughout the time that we have dedicated to preparing this book.

Tehran, Iran — Farhad Hosseinzadeh Lotfi
Qaemshahr, Iran — Ali Ebrahimnejad
Rasht, Iran — Mohsen Vaez-Ghasemi
Qazvin, Iran — Zohreh Moghaddas

Contents

Chapter 1
Introduction to Data Envelopment Analysis and Fuzzy Sets

Abstract Data Envelopment Analysis (DEA) is a mathematical based programing technique for performance evaluation of a set of Decision Making Units (DMUs). This technique is widely used for assessing different systems with different inputs and outputs in different fields. In this chapter some basic definition of mathematical modeling with DEA technique will be reviewed and introduced. The definitions of efficiency, relative efficiency, effectiveness, and productivity are given. It will be shown how basic DEA models on basis of different Production possibility sets (PPSs) can be formulated. Also, some characteristics of envelopment and multiplier models will be discussed. Finally, the main concepts of fuzzy set theory are introduced.

Keywords Data envelopment analysis (DEA) · Production possibility sets (PPSs) · Decision making units (DMUs) · Efficiency · Effectiveness · Productivity · Fuzzy sets

1.1 A Brief Review on Data Envelopment Analysis

To show the importance and effects of data envelopment analysis (DEA) technique in literature, some of the works about DEA mentioned in literature is reviewed in this section as follows.

Charnes et al. [1] provided a book about DEA technique, its theory, Methodology, and applications. They mentioned that, this book is the first book that includes a comprehensive review and comparative discussion of the basic DEA models. Moreover, they claimed that this is the first book that addresses the actual process of conducting DEA analyses. Färe and Grosskopf [2] provided a book for making improvements in the efficiency literature to the case of intertemporal models. In this book the static network models and dynamic models have been introduced. Cooper et al. [3] provided a book which is carefully designed for systematically introducing DEA models and its uses as an applicable method in order to assess problems in different fields. The authors have noted that DEA has

F. Hosseinzadeh Lotfi et al., *Data Envelopment Analysis with R*, Studies in Fuzziness and Soft Computing 386, https://doi.org/10.1007/978-3-030-24277-0_1

been widely used for performance evaluation in many different kinds of entities tied up with different activities in variety of contexts in different countries. They have mentioned that the important feature of DEA that makes it famous is that in many activities DEA provides possibilities for using in cases which have been resistant to other methods due to complex relations between multiple inputs and multiple outputs. Cooper et al. [4] provided a book for systematically presenting the basic principles and new developments in DEA. The authors have aimed at covering the understanding of the methodology and its uses and potentials. They covered the basic principles of DEA as well as more advanced treatments. Klemen [5] presented a book about measuring of returns to scale (RTS) using DEA. He noted that in management and politics the concept of returns to scale as well as scale efficiency have important and permanent consequences. Thus, on basis of the most common DEA models and their technology, Klemen [5] discussed about different methods and models to identify RTS status as well as their advantages and disadvantages. Cooper et al. [6] provided a book which covers important topics in DEA. This book describes the state of the field and extends the frontier of DEA research. The handbook's chapters divided into basic DEA models and DEA applications. Zhu and Cook [7] provided a book for presenting the methodology of DEA as a mathematical programming method and its models for assessing the performance of different decision making units. They noted that DEA has a strong link with production theory in economics and it has been widely used in variety of applications as a tool for benchmarking in operations management. Emrouznejad and Tavana [8] provided a book for introducing the developments and applications of fuzzy DEA (FDEA) which is now considered as an extension of the classic DEA. They noted that FDEA models are introduced for dealing with imprecise and ambiguous data in performance measurement problems. Atrash [9] provided a book for introducing robust statistical procedures for evaluating and measuring the relative efficiency of multiple decision making units using DEA technique. Wen [10] provided a book in which it is tried to gather methods and models for dealing with topics of uncertain DEA. Ozcan and Tone [11] presented a book for emphasizing the use of DEA in creation of optimization-based benchmarks within hospitals, physician group practices, and health care systems. Cook and Zhu [12] provided a book in order to improve the frontier of DEA research. They tried to provide a complete survey for the state-of-the art DEA modeling as well as network DEA. Blackburn et al. [13] provided a book in order to completely examine educational production and costs considering the nonparametric DEA technique. In this regard the authors have introduced DEA models and completely discussed about them.

Zhu [14] provided a book about the methodology of DEA as well as its utilization in performance evaluation and benchmarking under the context of multiple performance measures. Fare et al. [15] provided a book about advances in DEA that is concentrated on both theoretical developments as well as applications in performance evaluations. Ray et al. [16] provided a book for introducing the theoretical and methodological foundations of production efficiency analysis using benchmarking. Aparicio et al. [17] provided a book about the state of the art as well as recent advances in efficiency and productivity. Hwang et al. [18] provided a book

focusing on application of DEA in operations analytics that are significant tools and techniques for improving operation functions. Zhu [19] provided a book collecting the applications and empirical studies using DEA technique. Hosseinzadeh Lotfi et al. [20] provided a book for introducing DEA, its practical applications, and innovative developments. Tone [21] provided a book for assessing the performance of DMUs with special emphasis on forecasting models. This book can help researchers recognizing the most appropriate methodology for accurate decision makings. Novoa [22] mentioned that discussing about questions mooted in performance evaluation is resulted new models and methods in this field. In this occasion, DEA, as a management tool, provided the aggregation of financial and non-financial indicators into a single measure. Thus, Novoa [22] provided a book for studding DEA from two aspects, normative to descriptive performance evaluation. Sherman [23] provided a book discussed about measurement of hospital efficiency using DEA technique. Suzuki and Nijkamp [24] provided a book introducing a newly improved distance function minimization (DFM) model, based on DEA technique. They noted that the DFM model can produce a most effective solution in efficiency improvement projections while analyzing inefficient units. Also, this book provides a set of fresh contributions to a quantitative evaluation. Panik [25] provided a book for comprehensively addressing the shortcomings of other texts in comparison with linear programming theory specifically using DEA in resource allocation. Huber et al. [26] presented a book about Multiple Criteria Decision Making in which a comprehensive set of practical problems comprising multiple criteria, including numerous approaches are provided. The authors mentioned that, their book can be a good complement for traditional textbooks and lecture material. This book presents a comprehensive set of practical problems comprising multiple criteria, including numerous approaches for their solution, for decision support. Kahraman and Otay [27] provided a book for presenting a fuzzy multiple criteria decision making problem in neutrosophic sets which can be useful for considering certain types of uncertainty that classical methods could not take into account. Kao [28] provided a book dealing with network DEA, its developments and applications. He noted that network DEA models improved conventional DEA as they also consider the inter relation of the system. Moreover, the "network DEA" models can help recognize the specific components that contribute inefficiencies into the overall systems and reveal the relationships between the system and component efficiencies. Ebrahimnejad and Verdegay [29] provided a book offers a comprehensive introduction to the fuzzy mathematical programming (FMP) problems. Their book gives a complete introduction to the basic concepts, as well as extended information about computational-intelligence-based optimization models with a focus on fuzzy transportation problems. Sueyoshi and Goto [30] provided a book to develop methods and models while using DEA for evaluation of sustainability developments. They noted that their research is divided into two sections, the first considers classic DEA models and the second considers conceptual and methodological developments of classical DEA models in regards of environmental assessments. Khezrimotlagh and Chen [31] provided a book for introducing methods for performance evaluation of a set of units using DEA.

Paradi et al. [32] provided a book with the focus on the methodology and applications of DEA. The aim of this book is to consider firms dealing with financial services and using DEA method for measuring productivity, efficiency and effectiveness. On basis of fuzzy information Zhou and Xu [33] presented a book introducing some approaches for qualitative investment decision-making. The main approaches are based on the hesitant fuzzy information, the hesitant fuzzy trade-off and portfolio selection, the hesitant fuzzy preference envelopment analysis.

1.2 Basic Definitions

In this section, we describe the basic concepts of DEA such as efficiency, effectiveness and productivity.

One of the things that all organizations and even humans pay attention to is the issue of productivity. In each organization, productivity depends on two factors: efficiency and effectiveness. For this reason, we first explore the concept of efficiency and effectiveness.

Organizations always use combinations of facilities to achieve the predefined objectives, so that these products lead to goals and programs. For further details, consider the following example.

A university is created in a certain place. The objectives of this university can be solving the problems of society, training of specialist and committed staff, producing knowledge for the welfare of the community and etc. In beginning, each university uses the facilities such as human resources (faculty, staff and students), budgets, laboratories, and workshops, buildings, and more. To achieve its goals, the university tries to graduate students, publish scientific articles and books, and implement research projects. However, which of these products has been able to estimate the goals of the university is unclear. Symbolically, this process can be shown in Fig. 1.1.

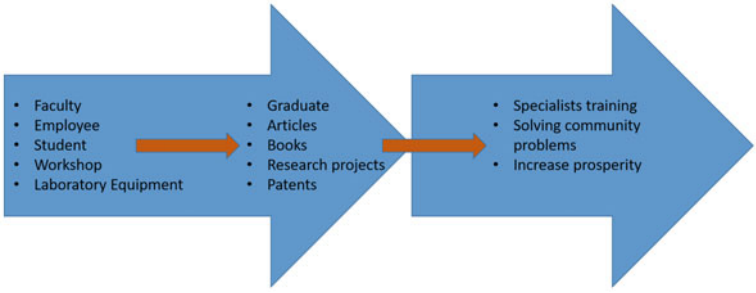

Fig. 1.1 Efficiency and effectiveness

Efficiency is defined as the function of the obtained results and the outputs produced by using the resources. Effectiveness is described in terms of how much work is done and how well the products fit into the organization's goals. How effective productions can be in society is defined as effectiveness, so the effectiveness is the function of products and objectives of the organization.

Leading organizations have always paid special attention to both efficiency and effectiveness indicators and try to measure both of them. It can be said that organizational productivity is in the best position whenever both efficiency and effectiveness indicators are optimal. The diagram given in Fig. 1.2 describes this relationship.

In most organizations, achieving goals can be a guarantee of the quality of the organization's products and thus brings credibility and branding. Therefore, when the effectiveness is high, an organization can hope for a good future. High performance compared to the facilities available to the organization can reflect the performance status of an organization in the current situation. Therefore, the efficiency and effectiveness interprets respectively the current and future conditions, and therefore, when the organization is productive, it has a good current situation and hopes for a great future.

The data envelopment analysis technique calculates the relative efficiency of a set of heterogeneous decision-making units (DMUs). In this technique, first, with the help of real observations of the society (n is the number of the observed DMUs), a production possibility set is created, and then in this set it is possible to search whether there is a better point than the DMU under consideration. If there is a better point, the unit will be inefficient and otherwise it will be efficient. For this purpose, we first consider a fix direction for improvement, and then try to improve the unit under evaluation in the production possibility set. Since the production possibility set is the smallest set that can be considered, the resulting relative efficiency is optimistic.

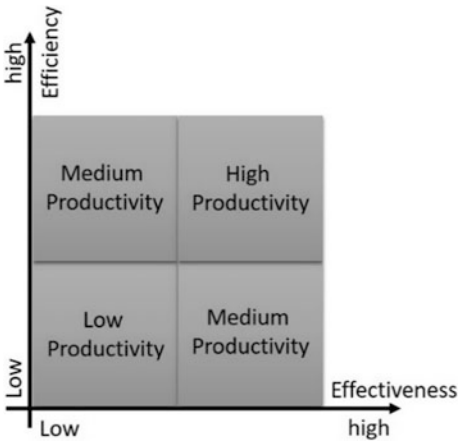

Fig. 1.2 Diagram of efficiency, effectiveness, and productivity

1.3 Different Models of DEA

In this section, we first provide the basic definitions in DEA. Then, we show how to model some of the basic DEA models in detail.

Suppose n decision making units (DMUs) are considered to be evaluated each of which uses m different inputs for producing s different outputs. Symbolically, a DMU can be shown as Fig. 1.3.

First, let X and Y be the input and output vectors, respectively, and assume $Y_j \neq 0, Y_j \geq 0$ and $X_j \neq 0, X_j \geq 0$.

The absolute efficiency of the DMU_p is defined as follows:

$$E_P = \frac{U^t Y_p}{V^t X_p} = \frac{\sum_{r=1}^{s} u_r y_{rp}}{\sum_{i=1}^{m} v_i x_{ip}} \tag{1.1}$$

The very first definitions of efficiency have been given by Farell [34] and then modified by Charnes et al. [35] and finally improved by Banker et al. [36].

In mode (1.1), v_i and u_r are respectively the weights of ith input and rth output in which $v_i > 0, u_r > 0$.

The relative efficiency of DMU_p is then defined as follows:

$$RE_P = \frac{E_P}{\text{Max}\{E_j : \quad j = 1, \ldots, n\}} = \frac{E_P}{\text{Max}\left\{\frac{U^t Y_j}{V^t X_j} : \quad j = 1, \ldots, n\right\}} \tag{1.2}$$

As the aim of DEA is deriving optimistic relative efficiency score, thus model (1.3) is formulated.

$$\begin{aligned} \text{Max} \quad & RE_P = \frac{E_P}{\text{Max}\left\{\frac{U^t Y_j}{V^t X_j}; \quad j=1,\ldots,n\right\}} \\ \text{s.t.} \quad & U > 0, \ V > 0 \end{aligned} \tag{1.3}$$

Using the variable transformation presented in Charnes et al. [35], model (1.3) can be converted into model (1.4).

$$\begin{aligned} \text{Max} \quad & RE_P = \frac{U^t Y_p}{V^t X_p} \\ \text{s.t.} \quad & \frac{U^t Y_j}{V^t X_j} \leq 1, \quad j = 1, \ldots, n \\ & U > 0, \ V > 0 \end{aligned} \tag{1.4}$$

Fig. 1.3 DMU_j, where $X_j = (x_{1_j}, \ldots, x_{m_j})$ is the input vector and $Y_j = (y_{1_j}, \ldots, y_{s_j})$ is the output vector of the decision making unit j

To be able to use optimization methods to solve the model (1.4), instead of constraints $u > 0$ and $v > 0$, constraints $U \geq \frac{1}{s}\varepsilon$ and $V \geq \frac{1}{m}\varepsilon$ should be used. In this case by using variable transformation, the following model will be obtained:

$$\begin{array}{ll} \text{Max} & RE_P = UY_o \\ \text{s.t.} & VX_o = 1, \\ & UY_j - VX_j \leq 0, \quad j = 1, \ldots, n, \\ & U \geq 1\varepsilon,\ V \geq 1\varepsilon. \end{array} \tag{1.5}$$

Model (1.5) is called input-oriented CCR multiplier model. Dual of this linear model is as follows.

$$\begin{array}{ll} \text{Min} & \theta - 1\varepsilon(S^- + S^+) \\ \text{s.t.} & \lambda X + S^- = \theta X_o \\ & \lambda Y - S^+ = Y_o \\ & \lambda \geq 0, \\ & S^- \geq 0,\ S^+ \geq 0. \end{array} \tag{1.6}$$

Vector forms of models (1.5) and (1.6) are as follows:

$$\begin{array}{lll} \text{Max} & RE_P = \sum_{r=1}^{s} u_r y_{ro} & \\ \text{s.t.} & \sum_{i=1}^{m} v_i x_{io} = 1, & \\ & \sum_{r=1}^{s} u_r y_{rj} - \sum_{i=1}^{m} v_i x_{ij} \leq 0, & j = 1, \ldots, n, \\ & u_r \geq 1\varepsilon,\ v_i \geq 1\varepsilon, & \text{r} = 1, \ldots, \text{s},\ \text{i} = 1, .., \text{m}. \end{array} \tag{1.7}$$

$$\begin{array}{lll} \text{Min} & \theta - 1\varepsilon\left(\sum_{i=1}^{m} s_i^- + \sum_{r=1}^{s} s_r^+\right) & \\ \text{s.t.} & \sum_{j=1}^{n} \lambda_j x_{ij} + s_i^- = \theta x_{io}, & i = 1, \ldots, m, \\ & \sum_{j=1}^{n} \lambda_j y_{rj} - s_r^+ = y_{ro}, & r = 1, \ldots, s, \\ & \lambda_j \geq 0, & j = 1, \ldots, n, \\ & s_i^- \geq 0,\ s_r^+ \geq 0. & i = 1, \ldots, m,\ r = 1, \ldots, s. \end{array} \tag{1.8}$$

Model (1.6) or its equivalent (1.8) is called input-oriented CCR envelopment two-phase model. According to strong duality theorem, the optimal objective values for a pair of bounded primal and dual problems are the same. If such pair of models have the efficiency scores equals to one, then the DMU under evaluation is called efficient otherwise it is called inefficient.

Here, θ^* shows the radial efficiency of DMUp. For inefficient DMUs it is possible to obtain target points (Benchmark) as follows:

$$Benchmark = (\sum_{j=1}^{n} \lambda_j^* x_j, \sum_{j=1}^{n} \lambda_j^* y_j) \tag{1.9}$$

One of the models (1.5) or (1.6) should be solved for each of the n decision making units in order to obtain the relative efficiency scores. Also, model (1.6) is used to find target units of inefficient DMUs.

Note that, the point of differentiation and the superiority of the data envelopment analysis technique on other methods of evaluation and ranking methods is the determination of resources and inefficiencies in each of the factors affecting performance.

In most applications, the relative importance of input and output indicators can be expressed by the decision maker as a trade-off between inputs and outputs. Trade-offs can be expressed as an axiom in DEA. The addition of the trade-offs axiom will expand the production possibility set, and therefore some efficient units may become inefficient. This trade-off axiom can be shown by the following relation:

$$UA^t + VB^t \leq 0 \tag{1.10}$$

Thus, adding this constraint to the basic classic models, CCR model with weight restrictions will be obtained as follows:

$$\begin{array}{lll} \text{Max} & RE_P = UY_o & \\ \text{s.t.} & VX_o = 1, & \\ & UY_j - VX_j \leq 0, & j = 1, \ldots, n, \\ & UA^t + VB^t \leq 0, & \\ & U \geq 1\varepsilon,\ V \geq 1\varepsilon. & \end{array} \tag{1.11}$$

Consequently, the CCR model with weight restrictions will be as follows:

$$\begin{array}{lll} \text{Min} & \theta & \\ \text{s.t.} & \sum_{j=1}^{n} \lambda_j x_{ij} - \sum_{j=1}^{n} \mu_j a_{ij} \leq \theta x_{io} & i = 1, \ldots, m \\ & \sum_{j=1}^{n} \lambda_j x_{ij} + \sum_{j=1}^{n} \delta_j b_{rj} \geq y_{ro} & r = 1, \ldots, s \\ & \lambda_j \geq 0, & j = 1, \ldots, n \\ & \delta_j \geq 0,\ \mu_j \geq 0, & j = 1, \ldots, n \end{array} \tag{1.12}$$

Note that DEA models are formulated and implemented according to the adopted axioms. The overall model for evaluation can be displayed as follows.

$$\begin{array}{ll} \text{Max} & \alpha \\ \text{s.t.} & \begin{pmatrix} X_p \\ Y_p \end{pmatrix} + \alpha \begin{pmatrix} -d^x \\ d^y \end{pmatrix} \in T \end{array} \tag{1.13}$$

Note that:

$$d^x \geq 0, \quad d^y \geq 0, \quad \begin{pmatrix} -d^x \\ d^y \end{pmatrix} \neq 0 \tag{1.14}$$

Here corresponding production possibility set is defined as:

$$T = \left\{ \begin{pmatrix} X \\ Y \end{pmatrix} | X \geq \sum_{j=1}^{n} \lambda_j x_{ij}, X \leq \sum_{j=1}^{n} \lambda_j y_{rj}, \quad \lambda \in \Lambda \right\} \tag{1.15}$$

Considering different Λ and d, variety of DEA models will be obtained as mentioned in Table 1.1.

Additive models are introduced by Bardhan et al. [37]. Färe and Grosskopf [38] introduced CCR-BCC and BCC-CCR models. FDH model is introduced by Deprins et al. [39].

Many DEA models are introduced in literature dealing with efficiency evaluations, classified into radial and non-radial models. Also, some models are

Table 1.1 Different DEA models

Λ	d	Model
$\Lambda = \{\lambda \mid \lambda \geq 0\}$	$d = \begin{pmatrix} -X_p \\ 0 \end{pmatrix}$	Input-oriented CCR model
$\Lambda = \{\lambda \mid \lambda \geq 0\}$	$d = \begin{pmatrix} 0 \\ Y_p \end{pmatrix}$	Output-oriented CCR model
$\Lambda = \{\lambda \mid \lambda \geq 0\}$	$d = \begin{pmatrix} -X_p \\ Y_p \end{pmatrix}$	Additive model in $\boldsymbol{T_c}$
$\Lambda = \{\lambda \mid \lambda \geq 0 \ \& \ \sum_{j=1}^{n} \lambda_j = 1\}$	$d = \begin{pmatrix} -X_p \\ 0 \end{pmatrix}$	Input-oriented BCC model
$\Lambda = \{\lambda \mid \lambda \geq 0 \ \& \ \sum_{j=1}^{n} \lambda_j = 1\}$	$d = \begin{pmatrix} 0 \\ Y_p \end{pmatrix}$	Output-oriented BCC model
$\Lambda = \{\lambda \mid \lambda \geq 0 \ \& \ \sum_{j=1}^{n} \lambda_j = 1\}$	$d = \begin{pmatrix} -X_p \\ Y_p \end{pmatrix}$	Additive model in $\boldsymbol{T_v}$
$\Lambda = \{\lambda \mid \lambda \geq 0 \ \& \ \sum_{j=1}^{n} \lambda_j \geq 1\}$	$d = \begin{pmatrix} -X_p \\ 0 \end{pmatrix}$	Input-oriented BCC-CCR model
$\Lambda = \{\lambda \mid \lambda \geq 0 \ \& \ \sum_{j=1}^{n} \lambda_j \geq 1\}$	$d = \begin{pmatrix} 0 \\ Y_p \end{pmatrix}$	Output-oriented BCC-CCR model
$\Lambda = \{\lambda \mid \lambda \geq 0 \ \& \ \sum_{j=1}^{n} \lambda_j \leq 1\}$	$d = \begin{pmatrix} -X_p \\ 0 \end{pmatrix}$	Input-oriented CCR-BCC model
$\Lambda = \{\lambda \mid \lambda \geq 0 \ \& \ \sum_{j=1}^{n} \lambda_j \leq 1\}$	$d = \begin{pmatrix} 0 \\ Y_p \end{pmatrix}$	Output-oriented CCR-BCC model
$\Lambda = \{\lambda \mid \lambda \geq 0 \ \& \ \lambda_j \in \{0,1\}, \quad \forall j, \sum_{j=1}^{n} \lambda_j = 1\}$	$d = \begin{pmatrix} -X_p \\ 0 \end{pmatrix}$	Input-oriented FDHV model

introduced for cost, revenue, and profit efficiency evaluations. Also, Malmquist productivity index used DEA models for assessing different DMUs in different time periods. Also, with DEA models it is possible to obtain directional efficiency scores. Ranking DMUs is also possible with using DEA models. Determining the status of returns to scale of efficient units is also possible with DEA.

1.4 Fuzzy Set Theory

In this subsection, we shall introduce some basic concepts of fuzzy sets, fuzzy numbers, fuzzy arithmetic and ranking fuzzy numbers [29, 40–47].

Definition 1.1 Let X be a universe of discourse. Then a fuzzy set $\tilde{A}$ in X is defined by the set

$$\tilde{A} = \left\{ \left(x, \mu_{\tilde{A}}(x)\right) | x \in X \right\}$$

where $\mu_{\tilde{A}} : X \rightarrow [0, 1]$ is called the membership function of fuzzy set $\tilde{A}$. The value of $\mu_{\tilde{A}}(x)$ represents the degree of membership of x being in $\tilde{A}$.

Definition 1.2 The support of a fuzzy set $\tilde{A}$ is denoted by $supp(\tilde{A})$ where $supp\,(\tilde{A}) = \left\{x \in X; \mu_{\tilde{A}}(x) > 0\right\}$.

Definition 1.3 A fuzzy set $\tilde{A}$ is called normal if there exists $x_\circ \in X$ such that $\mu_{\tilde{A}}\,(x_\circ) = 1$.

Definition 1.4 A fuzzy set $\tilde{A}$ on X is called a convex fuzzy set if $\mu_{\tilde{A}}(\lambda x_1 + (1 - \lambda)x_2) \geq \min\left\{\mu_{\tilde{A}}(x_1), \mu_{\tilde{A}}(x_2)\right\}$ whenever $x_1, x_2 \in X$ and $\lambda \in [0, 1]$.

Definition 1.5 Let $\tilde{A}_1, \tilde{A}_2, \ldots, \tilde{A}_n$ be n fuzzy subsets of $X_1, X_2, \ldots, X_n$, respectively. The Cartesian product of $\tilde{A}_1, \tilde{A}_2, \ldots, \tilde{A}_n$ is a fuzzy set on $X = X_1 \times X_2 \times \cdots \times X_n$ defined as follows:

$$\begin{aligned}&\tilde{A}_1 \times \tilde{A}_2 \times \cdots \times \tilde{A}_n \\ &= \left\{ \left((x_1, \ldots, x_n), \mu_{\tilde{A}_1 \times \tilde{A}_2 \times \cdots \times \tilde{A}_n}(x_1, \ldots, x_n) \right) | x_i \in X_i, i = 1, \ldots, n \right\}\end{aligned}$$

where

$$\mu_{\tilde{A}_1 \times \tilde{A}_2 \times \cdots \times \tilde{A}_n}(x_1, \ldots, x_n) = \min\left\{ \mu_{\tilde{A}_i}(x_i), i = 1, 2, \ldots, n \right\}$$

Definition 1.6 (*Extension principle* [43]) Let f be a mapping form $X = X_1 \times X_2 \times \cdots \times X_n$ to a universe Y, i.e., $f : X \rightarrow Y$, and $\tilde{A} = \tilde{A}_1 \times \tilde{A}_2 \times \cdots \times \tilde{A}_n$ be a fuzzy set on X. Then, $\tilde{B} = f(\tilde{A})$ is a fuzzy set on Y defined as follows:

$$\tilde{B} = f(\tilde{A}_1 \times \tilde{A}_2 \times \cdots \times \tilde{A}_n) = \left\{(y, \mu_{\tilde{B}}(y)), y = f(x_1, x_2, \ldots, x_n), x_i \in X_i, 1 \leq i \leq n\right\}$$

where

$$\mu_{\tilde{B}}(y) = \begin{cases} \sup\limits_{x: y = f(x_1, x_2, \ldots, x_n)} \min\limits_{1 \leq i \leq n} \left\{\mu_{\tilde{A}_i}(x)\right\} & f^{-1}(y) \neq \varnothing \\ 0 & f^{-1}(y) = \varnothing \end{cases}$$

Definition 1.7 The complement of a fuzzy set $\tilde{A}$ is denoted by $\tilde{A}^c$ where

$$\tilde{A}^c = \left\{\left(x, \mu_{\tilde{A}^c}(x)\right) \middle| x \in X, \mu_{\tilde{A}^c}(x) = 1 - \mu_{\tilde{A}}(x)\right\}.$$

Definition 1.8 The union of $\tilde{A}$ and $\tilde{B}$ is denoted by $\tilde{A} \cup \tilde{B}$ and is defined as a fuzzy set given by

$$\tilde{A} \cup \tilde{B} = \left\{\left(x, \mu_{\tilde{A} \cup \tilde{B}}(x)\right) \middle| x \in X, \mu_{\tilde{A} \cup \tilde{B}}(x) = \max\left\{\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)\right\}\right\}$$

Definition 1.9 The intersection of $\tilde{A}$ and $\tilde{B}$ is denoted by $\tilde{A} \cap \tilde{B}$ and is defined as a fuzzy set given by

$$\tilde{A} \cap \tilde{B} = \left\{\left(x, \mu_{\tilde{A} \cap \tilde{B}}(x)\right) \middle| x \in X, \mu_{\tilde{A} \cap \tilde{B}}(x) = \min\left\{\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)\right\}\right\}$$

Definition 1.10 The $\alpha-$ cut of fuzzy set $\tilde{A}$ is denoted by $\left[\tilde{A}\right]_\alpha$ that is defined as

$$\left[\tilde{A}\right]_\alpha = \begin{cases} \left\{x \in X \middle| \mu_{\tilde{A}}(x) \geq \alpha\right\}, & if \;\; 0 < \alpha \leq 1 \\ cl\left(supp\left(\tilde{A}\right)\right), & if \;\; \alpha = 0. \end{cases}$$

where $cl(T)$ means the closure of subset $T \subseteq X$.

Definition 1.11 A fuzzy set $\tilde{A}$ defined on the real line $\mathbb{R}$ is called a fuzzy number if

- $\tilde{A}$ is normal.
- $\mu_{\tilde{A}}$ is a upper semi-continuous function.
- $\tilde{A}$ is fuzzy convex.
- $\left[\tilde{A}\right]_0 = cl\left(supp\left(\tilde{A}\right)\right)$ is a compact set.

Thus, mathematically the membership function $\mu_{\tilde{A}}$ of a fuzzy number $\tilde{A}$ is of the following form [29, 41]:

$$\mu_{\tilde{A}}(x) = \begin{cases} l(x), & a \leq x \leq b, \\ 1, & b \leq x \leq c, \\ r(x), & c \leq x \leq d, \end{cases} \tag{1.16}$$

where $l(x)$ and $r(x)$ are piecewise continuous, increasing and decreasing functions in $[a, b]$ and $[c, d]$, respectively.

Definition 1.12 A fuzzy number $\tilde{A}$ is called positive (resp. negative) if its membership function is such that $\mu_{\tilde{A}}(x) = 0$ for all $x < 0$ (resp. $x > 0$).

Definition 1.13 [29, 43] A function $L : (-\infty, \infty) \to [0, 1]$ is said to be the reference function if it satisfies the following conditions:

(i) $L(x) = L(-x) \quad \forall x \in (-\infty, \infty)$,
(ii) $L(0) = 1$,
(iii) $L(\cdot)$ is non-increasing and upper semi-continuous on $[0, \infty)$,
(iv) Either $L(1) = 0$ or $\lim_{x\to\infty} L(x) = 0$.

Definition 1.14 [29, 43] A flat fuzzy number $\tilde{A}$, denoted by $\tilde{A} = (a_1, a_2, a_3, a_4)_{LR}$, is said to be an *LR* flat fuzzy number, if its membership function is given by

$$\mu_{\tilde{A}}(x) = \begin{cases} L\left(\frac{a_2 - x}{a_2 - a_1}\right), & a_1 \le x \le a_2, \\ 1, & a_2 \le x \le a_3, \\ R\left(\frac{x - a_3}{a_4 - a_3}\right), & a_3 \le x \le a_4. \end{cases} \tag{1.17}$$

where L and R are the reference functions of $\tilde{A}$.

Definition 1.15 A flat fuzzy number $\tilde{A}$ with the reference functions $L(x) = R(x) = \max\{0, 1 - |x|\}$ is called a trapezoidal fuzzy number.

Thus, mathematically the membership function $\mu_{\tilde{A}}$ of a trapezoidal fuzzy number $\tilde{A} = (a_1, a_2, a_3, a_4)$ is of the following form [29]:

$$\mu_{\tilde{A}}(x) = \begin{cases} \frac{x - a_1}{a_2 - a_1}, & a_1 \le x \le a_2, \\ 1, & a_2 \le x \le a_3, \\ \frac{a_4 - x}{a_4 - a_3}, & a_3 \le x \le a_4. \end{cases} \tag{1.18}$$

Remark 1.1 A fuzzy trapezoidal number $\tilde{A} = (a_1, a_2, a_3, a_4)$ is reduced to a triangular fuzzy number if $a_2 = a_3$.

Mathematically the membership function $\mu_{\tilde{A}}$ of a triangular fuzzy number $\tilde{A} = (a_1, a_2, a_3)$ is of the following form

$$\mu_{\tilde{A}}(x) = \begin{cases} \dfrac{x - a_1}{a_2 - a_1}, & a_1 \le x \le a_2, \\ \dfrac{a_3 - x}{a_3 - a_2}, & a_2 \le x \le a_3. \end{cases} \tag{1.19}$$

Remark 1.2 If $L(1) = 0$ and $R(1) = 0$, the *LR* flat fuzzy number $\tilde{A}$ is called a finite flat fuzzy number. If $\lim_{x\to\infty} L(x) = 0$ and $\lim_{x\to\infty} R(x) = 0$, it is called an infinite flat fuzzy number.

Definition 1.16 Two *LR* flat fuzzy numbers $\tilde{A} = (a_1, a_2, a_3, a_4)_{LR}$ and $B = (b_1, b_2, b_3, b_4)_{LR}$ are said to be equal, i.e. $\tilde{A} = \tilde{B}$ if, and only if, $a_1 = b_1$, $a_2 = b_2$, $a_3 = b_3$ and $a_4 = b_4$.

Definition 1.17 An *LR* flat fuzzy number $\tilde{A} = (a_1, a_2, a_3, a_4)_{LR}$ is said to be a non-negative *LR* flat fuzzy number if, and only if, $a_1 \geq 0$.

Remark 1.3 The $\alpha-$ cut of the *LR* flat fuzzy number $\tilde{A} = (a_1, a_2, a_3, a_4)_{LR}$ is the closed interval

$$\left[\tilde{A}\right]_{\alpha} = \left[\left(\tilde{A}\right)_{\alpha}^{L}, \left(\tilde{A}\right)_{\alpha}^{U}\right] = \left[a_2 - (a_2 - a_1)L^{-1}(\alpha), a_3 + (a_4 - a_3)R^{-1}(\alpha)\right] \quad (1.20)$$

Remark 1.4 The $\alpha-$ cut of the trapezoidal fuzzy number $\tilde{A} = (a_1, a_2, a_3, a_4)$ is the closed interval.

$$\left[\tilde{A}\right]_{\alpha} = \left[\left(\tilde{A}\right)_{\alpha}^{L}, \left(\tilde{A}\right)_{\alpha}^{U}\right] = [a_1 + (a_2 - a_1)\alpha, a_4 - (a_4 - a_3)\alpha] \quad (1.21)$$

Remark 1.5 The $\alpha-$ cut of the triangular fuzzy number $\tilde{A} = (a_1, a_2, a_3)$ is the closed interval.

$$\left[\tilde{A}\right]_{\alpha} = \left[\left(\tilde{A}\right)_{\alpha}^{L}, \left(\tilde{A}\right)_{\alpha}^{U}\right] = [a_1 + (a_2 - a_1)\alpha, a_3 - (a_3 - a_2)\alpha] \quad (1.22)$$

Definition 1.18 Let $\tilde{A} = (a_1, a_2, a_3, a_4)_{LR}$ and $\tilde{B} = (b_1, b_2, b_3, b_4)_{LR}$ be any *LR* flat fuzzy numbers and $\tilde{C} = (c_1, c_2, c_3, c_4)_{RL}$ be any *RL* flat fuzzy numbers. Then,

(i) $\tilde{A} + \tilde{B} = (a_1 + b_1, a_2 + b_2, a_3 + b_3, a_4 + b_4)_{LR}$,
(ii) $\tilde{A} - \tilde{C} = (a_1 - c_4, a_2 - c_3, a_3 - c_2, a_4 - c_1)_{LR}$.

Definition 1.19 Let $\tilde{A} = (a_1, a_2, a_3, a_4)$ and $\tilde{B} = (b_1, b_2, b_3, b_4)$ be two trapezoidal fuzzy numbers. Then,

(i) $\tilde{A} + \tilde{B} = (a_1 + b_1, a_2 + b_2, a_3 + b_3, a_4 + b_4)$,
(ii) $\tilde{A} - \tilde{B} = (a_1 - b_4, a_2 - b_3, a_3 - b_2, a_4 - b_1)$.

Definition 1.20 Let $\tilde{A} = (a_1, a_2, a_3)$ and $\tilde{B} = (b_1, b_2, b_3)$ be two triangular fuzzy numbers. Then,

(i) $\tilde{A} + \tilde{B} = (a_1 + b_1, a_2 + b_2, a_3 + b_3)$,
(ii) $\tilde{A} - \tilde{B} = (a_1 - b_3, a_2 - b_2, a_3 - b_1)$.

Definition 1.21 Let $\tilde{A} = (a_1, a_2, a_3, a_4)_{LR}$ and $\tilde{B} = (b_1, b_2, b_3, b_4)_{LR}$ be two non-negative *LR* flat fuzzy numbers. Then,

$$\tilde{A} \otimes \tilde{B} \approx (a_1 b_1, a_2 b_2, a_3 b_3, a_4 b_4)_{LR}$$
$$\tilde{A} \div \tilde{B} \approx (\frac{a_1}{b_4}, \frac{a_2}{b_3}, \frac{a_3}{b_2}, \frac{a_4}{b_1})_{LR}, b_1 > 0$$

Definition 1.22 Let $\tilde{A} = (a_1, a_2, a_3, a_4)_{LR}$ be any *LR* flat fuzzy numbers and k be a scaler. Then,

$$k\tilde{A} = \begin{cases} (ka_1, ka_2, ka_3, ka_4)_{LR}, & k \geq 0, \\ (ka_4, ka_3, ka_2, ka_1)_{RL}, & k < 0. \end{cases}$$

Ranking fuzzy numbers is a necessary step in many fuzzy DEA models. In fact, fuzzy DEA models encounter steps when it is required to rank efficiency scores or to convert fuzzy constraints and objective functions into crisp ones. The following definitions are useful to perform this task.

Definition 1.23 [43] Let $\tilde{A}$ be a fuzzy number and a be a real number. Then, the possibility, the necessity and the credibility measures of $\tilde{A} \geq a$ are respectively defined by

$$\pi(\tilde{A} \geq a) = \sup_{x \geq a} \mu_{\tilde{A}}(x) \tag{1.23}$$

$$Nec(\tilde{A} \geq a) = 1 - \pi(\tilde{A} < a) = 1 - \sup_{x < a} \mu_{\tilde{A}}(x) \tag{1.24}$$

$$Cr(\tilde{A} \geq a) = \frac{1}{2}\left[\pi(\tilde{A} \geq a) + Nec(\tilde{A} \geq a)\right] \tag{1.25}$$

Definition 1.24 [47] Let $\tilde{A}$ be a fuzzy number. Then, the expected credit of a fuzzy number $\tilde{A}$ is defined by

$$E(\tilde{A}) = \int_0^{\infty} Cr(\tilde{A} \geq a) da - \int_{-\infty}^{0} Cr(\tilde{A} \leq a) da \tag{1.26}$$

Definition 1.25 [4] Let $\tilde{A}$ and $\tilde{B}$ be two fuzzy numbers. Then, the possibility of dominance $\tilde{A}$ to $\tilde{B}$ is defined by

$$PD(\tilde{A} \succeq \tilde{B}) = \sup_{x \geq y} \min[\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(y)] \tag{1.27}$$

The average of possibility dominance of a fuzzy number to k fuzzy numbers is obtained as follows

$$EV\left(\tilde{A}_1 \succeq \tilde{A}_2, \ldots, \tilde{A}_k\right) = \frac{PD\left(\tilde{A}_1 \succeq \tilde{A}_2\right) + \cdots + PD\left(\tilde{A}_1 \succeq \tilde{A}_k\right)}{k-1} \tag{1.28}$$

Definition 1.26 [46] Ranking functions (defuzzification functions) are transformation functions $\Re : F(\mathbb{R}) \to \mathbb{R}$ which associate each fuzzy number with a real number and then use the ordering $\geq$ on the real line. Here $F(\mathbb{R})$ denotes the set of all fuzzy number defined on $\mathbb{R}$.

1.5 Conclusion

In this chapter we reviewed some basic definition of mathematical modeling with DEA technique. We provided the definitions of efficiency, relative efficiency, effectiveness, and productivity and formulated basic DEA models using different Production possibility sets. Also, we discussed some characteristics of envelopment and multiplier DEA models. Finally, we presented the basic definitions in fuzzy set theory to deal with uncertain data in DEA models. According to the given basic definitions of DEA technique, one can now able to distinguish various DEA models. Also, different properties of the mentioned models are now evident for readers. Also, readers are now informed with basic concepts of fuzzy data.

References

1. Charnes, A., Cooper, W.W., Lewin, A.Y., Seiford, L.M.: Data Envelopment Analysis: Theory, Methodology, and Applications. Springer (1995). ISBN-13: 978-0792394792
2. Färe, R., Grosskopf, S.: Intertemporal Production Frontiers: With Dynamic DEA. Springer (1996). ISBN-13: 978-0792397090
3. Cooper, W.W., Seiford, L.M., Tone, K.: Introduction to Data Envelopment Analysis and Its Uses: With DEA-Solver Software and References. Springer (2005). ISBN-13: 978-0387285801
4. Cooper, W.W., Seiford, L.M., Tone, K.: Data Envelopment Analysis: a Comprehensive Text with Models, Applications, References and DEA-Solver Software (2006). ISBN-13: 978-0387452814
5. Klemen, B.: Data Envelopment Analysis: Returns-to-Scale Measurement. VDM Verlag (2009). ISBN-13: 978-3639167146
6. Cooper, W.W., Seiford, L.M., Joe, Z.: Handbook on Data Envelopment Analysis. In: International Series in Operations Research & Management Science. Springer (2011). ISBN-13: 978-1441961501
7. Zhu, J., Cook, W.D.: Data Envelopment Analysis: Balanced Benchmarking. Create Space Independent Publishing Platform (2013). ISBN-13: 978-1492974796
8. Emrouznejad, A., Tavana, M.: Performance Measurement with Fuzzy Data Envelopment Analysis. In: Studies in Fuzziness and Soft Computing. Springer (2013). ISBN-13: 978-3642413711
9. Al Atrash, A.R.: Robust Data Envelopment Analysis Model: Theory and Application. LAP LAMBERT Academic Publishing (2013). ISBN-13: 978-3659434426

10. Wen, M.: Uncertain Data Envelopment Analysis . In: Uncertainty and Operations Research. Springer (2014). ISBN-13: 978-3662438015
11. Ozcan, Y.A., Tone, K.: Health Care Benchmarking and Performance Evaluation: an Assessment Using Data Envelopment Analysis (DEA). In: International Series in Operations Research & Management Science. Springer (2014). ISBN-13: 978-1489974716
12. Cook, W.D., Zhu, J.: Data Envelopment Analysis: a Handbook of Modeling Internal Structure and Network. In: International Series in Operations Research & Management Science. Springer (2014). ISBN-13: 978-1489980670
13. Blackburn, V., Brennan, S., Ruggiero, J.: Nonparametric Estimation of Educational Production and Costs Using Data Envelopment Analysis. In: International Series in Operations Research & Management Science. Springer (2014). ISBN-13: 978-1489974686
14. Zhu, J.: Quantitative Models for Performance Evaluation and Benchmarking: Data Envelopment Analysis with Spreadsheets. In: International Series in Operations Research & Management Science. Springer (2015). ISBN-13: 978-3319066462
15. Fare, R., Grosskopf, S., Margaritis, D.: Advances in Data Envelopment Analysis. World Scientific (2015). ASIN: B01GTUQG9I
16. Ray, S.C., Kumbhakar, S.C., Dua, P.: Benchmarking for Performance Evaluation: a Production Frontier Approach. Springer (2015). ISBN-13: 978-8132222521
17. Aparicio, J., Knox Lovell, C.A., Pastor, J.T.: Advances in Efficiency and Productivity. In: International Series in Operations Research & Management Science. Springer (2016). ISBN-13: 978-3319484594
18. Hwang, S., Lee, H.S., Zhu, J.: Handbook of Operations Analytics Using Data Envelopment Analysis. In: International Series in Operations Research & Management Science. Springer (2016). ISBN-13: 978-1489977038
19. Zhu, J.: Data Envelopment Analysis: a Handbook of Empirical Studies and Applications. In: International Series in Operations Research & Management Science. Springer (2016). ISBN-13: 978-1489976826
20. Hosseinzadeh Lotfi, F., Najafi, S.E., Nozari, H.: Data Envelopment Analysis and Effective Performance Assessment. In: Advances in Business Information Systems and Analytics. IGI Global (2016). ISBN-13: 978-1522505969
21. Tone, K.: Advances in DEA Theory and Applications: With Extensions to Forecasting Models. In: Wiley Series in Operations Research and Management Science. Wiley (2017). ISBN-13: 978-1118945629
22. Novoa, N.V.: Data Envelopment Analysis: From Normative to Descriptive Performance Evaluation. In: Europäische Hochschulschriften/European University Studies/Publications Universitaires Européennes. Peter Lang GmbH, Internationaler Verlag der Wissenschaften (2017). ISBN-13: 978-3631724491
23. Sherman, H.D.: Measurement of Hospital Efficiency Using Data Envelopment Analysis. Springer (2017). ISBN-13: 978-1376178579
24. Suzuki, S., Nijkamp, P.: Regional Performance Measurement and Improvement: New Developments and Applications of Data Envelopment Analysis. In: New Frontiers in Regional Science: Asian Perspectives. Springer (2018). ISBN-13: 978-9811091148
25. Panik, M.J.: Linear Programming and Resource Allocation Modeling. Wiley (2018). ISBN-13: 978-1119509448
26. Huber, S., Geiger, M.J., Almeida, A.T.: Multiple Criteria Decision Making and Aiding: Cases on Models and Methods with Computer Implementations. In: International Series in Operations Research & Management Science. Springer (2018). ISBN-13: 978-3319993034
27. Kahraman, C., Otay, İ.: Fuzzy Multi-criteria Decision-Making Using Neutrosophic Sets. In: Studies in Fuzziness and Soft Computing. Springer (2018). ISBN-13: 978-3030000448
28. Kao, C.: Network Data Envelopment Analysis: Foundations and Extensions. In: International Series in Operations Research & Management Science. Springer (2016). ISBN-13: 978-3319317168

29. Ebrahimnejad, A., Verdegay, J.L.: Fuzzy Sets-Based Methods and Techniques for Modern Analytics. In: Studies in Fuzziness and Soft Computing. Springer (2018). ISBN-13: 978-3319739021
30. Sueyoshi, T., Goto, M.: Environmental Assessment on Energy and Sustainability by Data Envelopment Analysis. In: Wiley Series in Operations Research and Management Science. Wiley (2018). ISBN-13: 978-1118979341
31. Khezrimotlagh, D., Chen, Y.: Decision Making and Performance Evaluation Using Data Envelopment Analysis. In: International Series in Operations Research & Management Science. Springer (2018). ISBN-13: 978-3319763446
32. Paradi, J.C., Sherman, H.D., Tam, F.K.: Data Envelopment Analysis in the Financial Services Industry: a Guide for Practitioners and Analysts Working in Operations Research Using DEA. In: Operations Research & Management Science. Springer (2018). ISBN-13: 978-3319888316
33. Zhou, W., Xu, Z.: Qualitative Investment Decision-Making Methods under Hesitant Fuzzy Environments. In: Studies in Fuzziness and Soft Computing. Springer (2019). ISBN-13: 978-3030113483
34. Farrell, M.J.: The measurement of productive efficiency. J. R. Stat. Soc. A **120**, 253–281 (1957)
35. Charnes, A., Cooper, W.W., Golany, B., Seiford, L.M., Stutz, J.: Foundations of data envelopment analysis for Pareto-Koopmans efficient empirical production functions. J. Econ. (Neth.) **30**, 91–107 (1985)
36. Banker, R.D., Charnes, A., Cooper, W.W.: Some models for estimating technical and scale inefficiencies in data envelopment analysis. Manag. Sci. **30**(9), 1078–1092 (1984)
37. Bardhan, I., Bowlin, W.F., Cooper, W.W., Sueyoshi, T.: Models and measures for efficiency dominance in DEA, part I: additive models and MED measures. J. Oper. Res. Soc. Jpn. **39**, 322–332 (1996)
38. Färe, R., Grosskopf, S.: Estimation of returns to scale using data envelopment analysis: a comment. Eur. J. Oper. Res. **79**, 379–382 (1994)
39. Deprins, D., Simar, L., Tulkens, H.: Measuring labor inefficiency in post offices. In: Marchand, M., Pestieau, P., Tulkens, H. (eds.) The Performance of Public Enterprises: Concepts and Measurements, North-Holland, Amsterdam (1984)
40. Zadeh, L.A.: Fuzzy sets. Inf. Control **8**(3), 338–353 (1965)
41. Klir, G.J., Yuan, B.: Fuzzy Sets and Fuzzy Logic. Theory and Applications. Prentice-Hall, PTR, Englewood Cliffs (1995)
42. Lai, Y.J., Hwang, C.L.: Fuzzy Mathematical Programming. Springer, Berlin (1992)
43. Dubois, D., Prade, H.: Fuzzy Sets and Systems Theory and Applications. Academic Press, New York (1980)
44. Zimmermann, H.J.: Fuzzy Sets, Decision Making and Expert Systems. Kluwer Academic Publishers, Boston (1987)
45. Bellman, R.E., Zadeh, L.A.: Decision making in a fuzzy environment. Manag. Sci. **17**(4), 141–164 (1970)
46. Ebrahimnejad, A., Verdegay, J.L.: A survey on models and methods for solving fuzzy linear programming problems. In: Kahraman, C., Kaymak, U., Yazici, A. (eds.) Fuzzy Logic in Its 50th Year. Studies in Fuzziness and Soft Computing, vol. 341. Springer, Cham (2016)
47. Liu, B., Liu, Y.K.: Expected value of fuzzy variable and fuzzy expected value models. IEEE Trans. Fuzzy Syst. **10**(4), 445–450 (2002)

Chapter 2
Introductions and Definitions of R

Abstract R is a mathematical and object-oriented open source programming language designed first for statistical computing. But, this software is powerful for dealing with optimization models. Variety of optimization moles such as linear, non-linear, integer, binary, and quadratic problems can be considered to be solved in this software. In this chapter some basic definition about R and required commands used in writing DEA models with R codes will be presented. All the commands are review sequentially according to their concepts. Also, for each command some numerical examples are also provided for clarifing the usage of commands for the readers.

Keywords R software · Basic commands and definitions · R functions

2.1 Preliminaries of R

R is a mathematical and object-oriented programming language that was first designed for statistical computing and data mining. This language is in fact the S version of the open source version. The R project was conducted in 1995 by the Department of Statistics at the University of Auckland by Robert Gentelman and Ross Aichka, and hence the name R, the S language except R, in the SPSS software (well-known statistical bundle) is also implemented. Although the S plus and R commands are very similar, they have distinct cores. This software is licensed under the GNU, General Public License, and is free to the Software Foundation. It is currently being developed by a group of statisticians called the "Software Core Team" R page at www.r-project.org.

Although R is often used for statistical computations, it can also be used for matrix computations, which is similar to software such as Octave and MATLAB. The programming language R covers a wide range of linear and nonlinear optimization models, integer and quadratic programming as well as classical statistical tests, time series analysis, classification and clustering, and has high graphical capabilities. In the R environment, C, C++, and Fortran can be connected and

F. Hosseinzadeh Lotfi et al., *Data Envelopment Analysis with R*, Studies in Fuzziness and Soft Computing 386, https://doi.org/10.1007/978-3-030-24277-0_2

invoked when the program is running, and expert users can directly change objects R by C code. The most important features of R can be graphical facilities for data analysis and drawing diagrams, simple and advanced programming language including conditional expressions, loops, recursive functions, and a strong set of computing operators, arrays and matrices, Software packages for data mining and analysis, storage, retrieval and manipulation of data, a multi-purpose library for analytical operations, and data mining libraries. As a result, special attention has been paid to the software. Among its features there are several packages that can be used to analyze different statistical and even non-statistical analyzes. Several software packages for solving optimization models of data envelopment analysis are presented in this software. Researchers can also use the packages provided to solve their models. It should be noted that these programs cannot be modified and only consider the same model. In R, there are several packages for solving models of data envelopment analysis and different kinds of optimization models that help researchers in programming to solve their problems. Types of instructions for optimizing problems with linear and nonlinear objective functions and constraints (equality and inequality forms) are introduced in this programming language. There are also commands for solving quadratic optimization models. In the introduced instructions, there are some commands for solving optimization models with integer and binary variables.

2.2 Basic Definitions

These exist many books about learning R. In this section some basic definitions and commands used in writing R codes for DEA models, will be introduced. The definitions are from, Davies [1], Matloff [2], Cotton [3], Kabacoff [4], Zumel and Mount [5], Lander [6], Vries and Meys [7], Alvaro [8], and Adler [9].

As a powerful tool R is used in variety of fields. Kleiber and Zeileis [10] used R in applied economics. Petris and Petrone [11] provided a book for introducing dynamic Linear Models with R Cowpertwait and Metcalfe [12] provided a book for introduction of time series with R. Everitt and Hothorn [13] presented a book which introduces applied multivariate analysis with R. Hart et al. [14] provided a book for Pyomo—Optimization modeling in Python. Ledolter [15] presented a book for data mining and business analytics with R. Ramachandran and Tsokos [16] provided a book for mathematical statistics with applications in R. Bloomfield [17] provided a book for using R in numerical analysis in science and engineering. Crawley [18] provided a book for statistical analysis which introduced how to use R. Paulo [19] provided a book for modern optimization with R. Emilio [20] presented a book discussing about quality control with R. Taveras [21] provided a book about how R can be used by Excel users. Pfaff [22] presented a book for financial risk modelling and portfolio optimization with R. James et al. [23] presented a book for learning R with an introduction to statistical analysis. Yao [24] provided a book for learning R in 8 h. In useful manuals about introduction to R, internals, writing R extensions,

language and environment for statistical computing, R language definition, and R data import/export are shared with users by the R core team [25–30].

At the beginning of working with R software, it is necessary to consider the following points.

www.r-project.org

- R software can be downloaded free of charge and installed from the following website.

  www.r-project.org

- After installing R software, you can run the library () command to see all the software packages that you have assessed to them. At the beginning of work, about 25 software packages have been activated as an instances.

With the advent of knowledge new software packs are adapted to new ways, you can access the new software packages by visiting the following site.

http://cran.r-project.org

In this regard, you must first connect to the Internet, then select your package in the package menu to install it.

- To use the new installed software packages, you should first call the required package at the beginning of code using the library (name of package) command. You can also use the search () command to search for software packages that are loaded.
- Get started with R software, a window called R console is open. To write the required program you need the R Editor window that can be selected from the new script option from the file menu.

All R outputs except graphs are displayed in the R console window. It should be noted that one can also use this window to get familiar with the functions and calculations. Each line in the R console environment begins with the ">" sign, and until the phrase is completed, each line begins with the "+" sign.

- The R software differs between uppercase and lowercase letters. Also R does not consider the distance. For example, typing 2 plus 3 (2 + 3) by typing the distance between them (2 + 3) has the same answer of 5.
- Since there may be same names running in the several program, it is necessary to use the rm () command after each run of the program in order to delete the previous values from the R memory.

Also one can use the shortcut Ctrl+L in order to delete the contents of the R console window. Also, rm (x, y) clears both x and y values.

- To use Help in R you can type the word after "?" sign in the R console, which is the equivalent of the command search help (). We also use apropos () to search for function names. You can also use the search term after the symbol "??". So by using these two commands, all the functions that have the phrase you are looking for can be identified. Also, use the help.search () function to search. You can also use the help.start () command to use the offline guide.

- The # sign is used to add descriptive phrases in the program text. Any text after this symbol is not read in the program.
- In the R program, we need to read from a file or write in a file to the default directory, which is called directory and working file.

To get the address of that file, simply use the getwd () command. You can also use the setwd(new address) command to change its address.

> "C:\desktop\R"

Suppose there is a working folder named R on the Desktop. By typing getwd () in the R console window and pressing Enter, we will have the following statement.

> "C:\desktop\R"

To change the working folder to the address "D: \ Newprog \ R", simply type the following command in Rconsole.

> setwd ("D:\Newprog\R")

Note when you exit the application and return to it again, the new address returns to the default address, so this setting is temporary. You can also select the 'set working directory' from the 'session' in the menu and with the help of that, select your working folder The dir () command can be used to see all the names of the files in the working folder.

- str () is one of the most used commands in R which is used to get a brief familiarity to each object or syntax in R.

With the help of the ls () command, you can list the names of the objects in the memory.

You can also use ls () to search for objects with special characters in their names.

- Also, summary ("filename") command can be used when you want to notify the contents of the file you entered to R
- The options () command can be used to change the number of outputs of different functions. As an example

$$> options(digits = 15)$$

Consider 15 decimal places to show the result. The default number for the digits is 7, and the maximum is 22.

Also, One can use round () command.

$$> round(\mathrm{x}, 3)$$

Consider 3 decimal places to show the result of x.

- To assign a name to an object, you can use the "=", "<-", "$\ll$ -", "->", and "- $\gg$" signs.

- To introduce variables, you can combine letters, numbers, and symbol "." and "-". It is important to note that the names of variables (objects) should not start with numbers and symbols such as "-".
- To get out of R, you can use the command.

2.3 Definition of Different Variables Types

In R, the following types of variables or objects can be defined.

- Character: X = "A"
- Numeric: *X* = "2"
- Integer: If x = 3, then R assumes it is numerical, but if x = 3L, then assumes it is integer.
- Complex: X = 1 + 4i
- logical: It can be either FALSE (F) or TRUE (T).

2.4 Attributes

The attributes of variables that can be considered in R are:

- Names
- 2-Dim: It is considered for vectors and matrixes.
- 3-Class: It specifies what type of variable is (for example, a character, a number, or ….).
- 4-Length: It is used for vectors.

Example 2.1 Consider the following example to verify and change the class of an object.

$$
\begin{aligned}
&x = 0:4\\
&x\\
&[1] \quad \circ \quad 1 \quad 2 \quad 3 \quad 4\\
&class(x)\\
&[1] \quad \text{"integer"}
\end{aligned}
$$

Now the class of x is convert to a number.

$$
\begin{aligned}
&> y = as.numeric\,(x)\\
&> class(y)\\
&[1] \quad \text{“numeric”}
\end{aligned}
$$

Note that the values of x and y are the same.

$$
\begin{aligned}
&> x\\
&[1] \quad \circ \quad 1 \quad 2 \quad 3 \quad 4\\
&> y\\
&[1 \quad \circ \quad 1 \quad 2 \quad 3 \quad 4
\end{aligned}
$$

2.5 Data Storage

In R, data can be stored in Vectors, Matrices, Arrays, Lists, and Data frame.

2.5.1 Vectors

Use the C () command to create a vector in R. In this way numbers can be put together to form a vector.

Example 2.2 Introduce the vector x.

$$
\begin{aligned}
&> x = c(4, 5, -3, 2, 1)\\
&[1] \quad 4 \quad 5 \quad -3 \quad 2 \quad 1
\end{aligned}
$$

To refer to a particular component of the vector, it is enough to write the number of that component inside [ ] along the name of the vector.

Example 2.3 Some **c**omponents of x vector are as follows:

$$
\begin{aligned}
&> x[2]\\
&[1] \quad 5\\
&> x[C(1, 4)]\\
&[1] \quad 4 \quad 2
\end{aligned}
$$

The length () command is used to specify the vector length.

Example 2.4 The length of vector x is:

$$\begin{array}{l} > length(x) \\ [1] \quad 5 \end{array}$$

Example 2.5 Suppose the vector x is 6 in length as follows.

$$> x = c(-1, 3, 2, 4, 7, 9)$$

The display of all vector elements except for the first and fourth elements is:

$$\begin{array}{l} > x[-c(1, 4)] \\ [1] \quad 3 \quad 2 \quad 7 \quad 9 \end{array}$$

To change the value of the first element, use the following command.

$$\begin{array}{l} > x[1] = -2 \\ > x \\ [1] \quad -2 \quad 3 \quad 2 \quad 4 \quad 7 \quad 9 \end{array}$$

Example 2.6 To get the conditional elements of a vector, we do as follows.

$$\begin{array}{l} > B = x < 5 \\ > B \\ [1] \quad T \quad T \quad T \quad T \quad F \quad F \\ > x[(x < 4) \& (x > 1)] \\ [1] \quad 3 \quad 2 \end{array}$$

Example 2.7 In this example names of elements for the vector x, is defined:

$$\begin{array}{l} > x = c(x = 3, y = 4, z = -3, w = 5) \\ > x \\ [1] \quad x \quad y \quad z \quad w \\ \quad\quad 3 \quad 4 \quad -3 \quad 5 \end{array}$$

Or as following:

$$\begin{array}{l} > x = c(1, 4, 3, 5, 7) \\ > names(x) = (\text{“}x\text{”}, \text{“}y\text{”}, \text{“}z\text{”}, \text{“}w\text{”}, \text{“}o\text{”}) \\ > x \\ [1] \quad x \quad y \quad z \quad w \quad o \\ \quad\quad 1 \quad 4 \quad 3 \quad 5 \quad 7 \end{array}$$

The following command can be used to define sequences of numbers by vector.

Example 2.8 The sequence of natural numbers from 20 to 30.

$$
\begin{array}{l}
> c(20:30) \\
[1] \quad 21 \quad 22 \quad 23 \quad 24 \quad 25 \quad 26 \quad 27 \quad 28 \quad 29 \quad 30
\end{array}
$$

You can also use the seq (from, to, by=) command to create an arithmetic progression. For example, the arithmetic progression between 1 and 15 with a ratio of 3 is defined as follows.

Example 2.9 Consider the following example.

$$
\begin{array}{l}
< seq(1, 15, by = 3) \\
[1] \quad 1 \quad 4 \quad 7 \quad 10 \quad 13
\end{array}
$$

The rep () command can be used to create a repeat in a vector.

Example 2.10 Consider the example of rep () command.

$$
\begin{array}{l}
> rep(1:5, 2) \\
[1] \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \\
> rep(1:2, each = 3) \\
[1] \quad 1 \quad 1 \quad 1 \quad 2 \quad 2
\end{array}
$$

The following command can be used to repeat non-numeric values.

Example 2.11 Consider the example for repeating non-numeric values

$$
\begin{array}{l}
> rep(3, 2, labels = c(\text{“}x\text{”}, \text{“}y\text{”}, \text{“}z\text{”})) \\
[1] \quad xx \quad yy \quad zz
\end{array}
$$

The first value (3) represents the number of levels. The second variable (2) represents the number of repetitions. Labels specifies the names of the levels.

2.5.2 Matrixes

The command matrix () creates a matrix from the given set of values.

$$
\begin{array}{l}
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, \\
\quad dimnames = NULL)
\end{array}
$$

Also, it is possible to turn a set of values into a matrix or test if the argument is a matrix or not using the following commands.

$$as.matrix()$$
$$is.matrix()$$

Any matrix can be filled either by row or by column.

Example 2.12 Creating a matrix.

$$> H = matrix(1:6, ncol = 3)$$
$$> H$$

	[,1]	[,2]	[,3]
[1,]	1	3	5
[2,]	2	4	6

Given the above example, it is clear that numbers 1 to 6 are in the form of a column in the matrix H. The ncol statement represents the number of columns. If we want the numbers to be placed in the matrix line, we use the byrow = T statement. The letter T stands for the word TRUE.

Example 2.13 Creating a matrix.

$$> H = matrix(1:g, ncol = 3, byraw = TRUE)$$
$$> H$$

	[,1]	[,2]	[,3]
[1,]	1	2	3
[2,]	4	5	6
[3,]	7	8	9

Example 2.14 To refer to the specific elements, columns, and rows of the matrix, one can act as follows.

$$> H[1,1]$$
$$[1] \quad 1$$
$$> H[3,2]$$
$$[1] \quad 8$$

All rows corresponding to all columns except the first and second columns are:

$$> H[,-c(1,2)]$$
$$[1] \quad 3 \quad 6 \quad 9$$

To specify elements that are larger than 5, the following command can be used.

$$> H[H > 5]$$
$$[1] \quad 6 \quad 7 \quad 8 \quad 9$$

To specify the specific rows and columns of the matrix, the following command can be used.

$$
\begin{array}{l}
> H[c(1,2), c(2,3)] \\
[1] \\
\qquad\quad [,1] \quad [,2] \\
[1,] \quad 2 \quad\quad 3 \\
[2,] \quad 5 \quad\quad 6
\end{array}
$$

You can also replace certain elements of matrices.

$$> H[1,3] = 7$$

In this case, by calling the H matrix, the number 7 will be replaced in the first row and the third column. Calling the first line of the matrix H we can see this change.

$$
\begin{array}{l}
> H[1,] \\
[1] \quad 1 \quad 2 \quad 7
\end{array}
$$

As you can see, the number 7 in the third column replaces the value of 3.

To combine vectors or matrices, rbind () and cbind () commands can be used to create a new combined matrix. The rbind and cbind functions combine matrices or vectors in columns and rows, respectively.

$$
\begin{array}{l}
cbind(\ldots, deparse.level = 0) \\
rbind(\ldots, deparse.level = 0)
\end{array}
$$

Note that in case of deparse.level = 0, no labels are considered, by default.

Example 2.15 Consider the following example where rbind () and Cbind () commands are used.

$$
\begin{array}{l}
> cbind(A = 1:4, B = 5:8, C = 9:12) \\
> \qquad\quad A \quad B \quad C \\
\qquad [1,] \quad 1 \quad 5 \quad 9 \\
\qquad [2,] \quad 2 \quad 6 \quad 10 \\
\qquad [3,] \quad 3 \quad 7 \quad 11 \\
\qquad [4,] \quad 4 \quad 8 \quad 12 \\
> x = 2:4; y = 3:5 \\
> rbind(x, y) \\
> \quad [,1] \quad [,2] \quad [,3] \\
x \quad\; 2 \quad\quad 3 \quad\quad 4 \\
y \quad\; 3 \quad\quad 4 \quad\quad 5
\end{array}
$$

Example 2.16 To name the rows and columns of a matrix, consider the following command.

$$
\begin{array}{l}
> \dim names(H) = list(c(\text{“}a_1\text{”}, \text{“}a_2\text{”}, \text{“}a_3\text{”}), c(\text{“}b_1\text{”}, \text{“}b_2\text{”}, \text{“}b_3\text{”})) \\
> H \\
\begin{array}{cccc}
 & b_1 & b_2 & b_3 \\
a_1 & 1 & 2 & 7 \\
a_2 & 4 & 5 & 6 \\
a_3 & 7 & 8 & 9
\end{array}
\end{array}
$$

Note that a matrix can also be defined as mentioned above.

Useful functions for naming the rows and columns of a matrix of the function rownames () and colnames ().

Example 2.17 Consider the following example.

$$
\begin{array}{l}
> x = matrix(1:4, nrow = 2) \\
> rownames(x) = LETTERS[1:2] \\
> colnames(x) = c(\text{“}a\text{”}, \text{“}b\text{”}) \\
> x \\
\begin{array}{ccc}
 & a & b \\
A & 1 & 3 \\
B & 2 & 4
\end{array}
\end{array}
$$

In the above example, the character vector of the LETTERS is a variable defined in R, which contains uppercase letters from A to Z. A vector consists of lowercase letter is defined with “letters”.

Example 2.18 Consider the following example. In this example, another method for naming the rows and columns of a matrix is introduced.

$$
\begin{array}{l}
> \dim names(x) = list(paste(\text{“}row\text{”}, LETTERS[1:2]), \\
> paste(\text{“}col\text{”}), LETTERS(1:2)] \\
> x \\
\begin{array}{ccc}
 & colA & colB \\
rowA & 1 & 3 \\
rowB & 2 & 4
\end{array}
\end{array}
$$

Use the paste () command to control the connection of strings together. The number of components that this function accepts is infinite. This function defaults an empty space between strings. You can use the sep argument to put the arbitrary character between the strings. If the data is vector-based, you should use the collapse argument instead of using sep.

$$paste(\ldots, sep = \text{“ ”}, collapse = NULL)$$

Example 2.19 Consider the following examples of the paste () command.

$$
\begin{aligned}
&> paste(\text{“}r\text{”}, 1:3, sep = \text{“.”})\\
&[1] \quad \text{“}r.1\text{”} \quad \text{“}r.2\text{”} \quad \text{“}r.3\text{”}\\
&> paste(\text{“}r\text{”}, 1:3, collapse = \text{“,”})\\
&[1] \quad \text{“}r_1, r_2, r_3\text{”}\\
&> x = 1:12\\
&> \dim(x) = c(3,4)\\
&> x
\end{aligned}
$$

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	1	4	7	10
[2,]	2	5	8	11
[3,]	3	6	9	12

Using the dim () command, we specify the number of rows and columns of a matrix (or in other words dimension of a matrix).

The following commands can be used to perform calculations on matrices.

diag (): Getting the original matrix diameter and constructing a diagonal matrix.
eigen (): Getting the eigen values of a matrix.
t (): Calculate the transpose of a matrix.
det (): Calculate the determinate of a matrix.
gin (): Calculates the inverse of a matrix.
sum (diag ()): Calculate the trace of a matrix (similar as trace (A) command).

Need to be noted that multiplication sign “% *%” used in order to multiply two matrices.

Notice the regular multiplication sign “*” does not multiply two matrixes, but multiplies only correspondence components of two matrixes.

2.5.3 Arrays

Arrays can be considered as a generalization of matrices. It can be said that each array has several matrices that are arranged below, and the number of matrices is the third dimension of the array.

$$array(data = NA, \dim = length(data), \dim names = NULL)$$

Also, the following commands can be used for testing an argument to be an array or considering an argument as an array.

$$is.array(x)$$
$$as.array(x, \ldots)$$

Example 2.20 Creating an array.

```
> array(1:6, dim(1:3))
[1]
     ,      , 1
   [1,]   [,1]   [,2]
             1      2
     ,      , 2
             [,1]   [,2]
   [1,]      3      4
     ,      , 3
             [,1]   [,2]
   [1,]      5      6
```

In the above example consider that if the second dimension is fixed, then we will have a matrix.

2.5.4 Lists

Note in vectors and matrices all layouts should be of the same type, but this is not the limit in the lists.

$$list(x, all.names = T, sorted = F, \ldots)$$

Also, the following commands can be used for testing an argument to be a list or considering an argument as a list.

$$is.list(x)$$
$$as.list(x, \ldots)$$

Example 2.21 Consider the following example.

```
> x = 1:2, y = c("a", "b")
> list.1 = list(x, y)
```

Note Use the [[]] symbol to specify the components of the list.

$$
\begin{array}{l}
> [[1]] \\
[1] \quad 1 \quad 2 \\
> [[2]] \\
[1] \quad \text{“}a\text{”} \quad \text{“}b\text{”}
\end{array}
$$

You can also call an arbitrary element from a list.

$$
\begin{array}{l}
> list.1[[1]] \\
[1] \quad 1 \quad 2 \\
> list.1[[2]] \\
[1] \quad \text{“}a\text{”} \quad \text{“}b\text{”}
\end{array}
$$

If the elements of a list have names, then in order to refer to a particular element the list $ name command can be used.

Example 2.22 In this example a particular element of a list is referred to.

$$
\begin{array}{l}
> list.1 = list(X = x, Y = y) \\
> list.1 \\
\$X \\
[1] \quad 1 \quad 2 \\
\$Y \\
[1] \quad \text{“}a\text{”} \quad \text{“}b\text{”}
\end{array}
$$

Note The $ symbol is used to derive a vector or component from a list or data frame.

2.5.5 Data Frames

The data frame can be referred to as a combination of lists and matrices. Note, the elements of the data frames can be of different types like lists, and the same as matrices the variables must have the same dimensions.

$$
\begin{array}{l}
data.frame(\ldots, row.names = NULL, check.rows = FALSE, \\
\quad check.names = TRUE, fix.empty.names = TRUE, \\
\quad stringsAsFactors = default.stringsAsFactors())
\end{array}
$$

Example 2.23 Consider the following example:

$$
\begin{aligned}
&> EFF = C(0.5, 0.37, 0.42, 0.77)\\
&> DMU = C(\text{“}a\text{”}, \text{“}b\text{”}, \text{“}c\text{”}, \text{“}d\text{”})\\
&> X = C(1, 7, 12, 13)\\
&> Y = C(2, 3, 5, 10)\\
&> data.frame(DMU, X, Y, EFF)\\
&[1]
\end{aligned}
$$

	DMU	*X*	*Y*	*EFF*
1	*a*	4	2	0.5
2	*b*	∘	3	0.37
3	*c*	12	5	0.42
4	*d*	13	10	0.77

as.data.frame(x, optional = TRUE,...) changes each argument of x to a data frame.

2.6 Mathematical Operators

The +, - and * symbols represent the addition, subtraction, and multiplication functions.

Also, /, ^ represents the division and power, respectively.

You can use functions % /% and %% to get a division and modulus.

2.7 Logical Operators

The symbols “<” and “>” are much larger and much smaller.

The symbols “<=” and “> =” denote larger, equal, and smaller equal.

“==” and “! =” reference to equality and inequality of the two components.

Symbol “!” represents the contradiction, and 1 and | represent “and” and “or” are logical AND and OR.

2.8 Use R in Calculation

You can use R to perform computational operations.

Example 2.24 Consider the following instances of computational operations.

$$
\begin{array}{l}
> 4+5 \\
[1] \quad 9 \\
> 4/5 \\
[1] \quad 0.8
\end{array}
$$

Example 2.25 Consider the following instances of computational operations. In the example below, the "Inf" note is infinite and "Nan" also means "Not A Number", that is, the amount that is not available.

$$
\begin{array}{l}
> 1/\circ \\
[1] \quad Inf \\
> -1/\circ \\
[1] \quad Inf \\
> \% \\
[1] \quad Nan
\end{array}
$$

"Inf" and "Nan" are specific values in R. Note that "NA" has been lost as it was previously described.

Example 2.26 Considering the vector x, $sqrt(X)$ is as follows.

$$
\begin{array}{l}
> X = C(2, 3, -1) \\
> sqrt(X) \\
[1] \quad 1.4 \quad 1.7 \quad NA
\end{array}
$$

Example 2.27 Creating a null vector.

$$
\begin{array}{l}
> C(\,) \\
[1]Null
\end{array}
$$

NULL means the collection is empty.

In R, we can assign a specific value to the variables by using the following command.

Example 2.28 In this example a specific value is assigned to a variable.

$$
\begin{aligned}
&> X = 2\\
&> X\\
&[1] \quad 2\\
&> Y = -4\\
&> x + Y\\
&[1] \quad -2
\end{aligned}
$$

2.9 Basic Mathematical Functions

Consider the following Table 2.1 including functions and their mathematical definitions.

Example 2.29 Consider the following instances for mathematics function.

Table 2.1 Mathematics function

Function	Mathematical definition
Abs	Abstract
Sum	Sum
Prod	Multiply
Sign	The sign function
Cum sum	Cumulative sum of vector numbers
Cum prod	Cumulative multiplication of a vector number
Which min	Place of min element
Which max	Place max
Average	Average
Median	Mead point
Var	Variance
Log	Log normal
Floor	Correct component
Exp	Exponential function
Which	Position of the element
Factorial	Factorial
Log10	Log at base 10
Round	Rounding the numbers
Sqrt	Square root
Trunc	The nearest integer to the number
Sin, cos	Trigonometric functions
Min, max	Minimum and maximum
Which	Zero position

$$
\begin{aligned}
&> abs(3-5)\\
&[1] \quad 2\\
&> sum(1:5)\\
&[1] \quad 15\\
&> floar(3,1)\\
&[1] \quad 3\\
&> factorial(3)\\
&[1] \quad 6\\
&> \log(,)\\
&[1] \quad Inf\\
&> round(sqr^{-1}(2)/4)\\
&[1] \quad 1/4142\\
&> \cos(2*pi)\\
&[1] \quad 1\\
&> X = C(10,2,5,2,4,2)\\
&> which(X=2)\\
&[1] \quad 2 \quad 4 \quad 6
\end{aligned}
$$

2.10 If Structure

The conditional clause if () is defined as follows.

$$if(cond)\{expr\}$$

If the statement (i.e. cond) is true, the result (i.e. expr) is executed.

Example 2.30 Consider the following example.

$$
\begin{aligned}
&> X = 3 \quad ; \quad Y = 5\\
&> if \quad (X<Y) \quad Z = X+1\\
&> Z\\
&[1] \quad 4
\end{aligned}
$$

Note that there is no need to write {} if the result of a condition is only one statement as expr in the defined command.

Example 2.31 Consider the following example.

```
> X = 2   ;   Y = 4
> if(X, Y)   &   Y + X = 6)   {
> X = X + 1
> Y = Y * 2
>    }
>    X
[1]  3
> Y
[1]  8
```

2.11 If Conditional

The structure of the if conditional command is as follows.

If (condition) {statement 1} else {statement 2}

$$if(cond)\{cons.expr\} \quad else \quad \{alt.expr\}$$

If the condition of the logical expression is true then the condition result (cons.expr) is executed, and if this statement is incorrect, then the else is run as the second statement (alt.expr).

Note that there is no need to write {} if the result of a condition is only one statement.

2.11.1 Example: Consider the Following Example

```
> Y = sqrt(4, 1, 9, 16)
> if(mean(Y) > 5)   Y = 1   else   Y = 2
> print(Y)
[1]  1
> if(mean(Y) > 5){y = 1, y = y + 2}else{y = 2, y = y + 3}
> print(Y)
[1]  3
```

2.12 If Else Conditional

Another condition command is if else (). This command contains three arguments.

$$if\ else\,(\text{test}, yes, no)$$

In this command the first argument is considered firstly. If it is true, the phrase "Yes" and if it is false, then "No" is performed without it.

Example 2.32 Consider the if else command as follows.

$$\begin{array}{l} > if\ else(Y > 2, NA, Y) \\ [1] \quad NA \quad NA \quad 3 \quad 4 \end{array}$$

If $Y > 2$ does not hold true, then it returns its own Y's values, and if $Y > 2$ then it returns the value of NA.

2.13 For Function

The other command is defined as the for () function, which is defined as follows.

$$For(\text{variable} \quad in \quad start : stop)\{\ \}$$

Example 2.33 Consider the following example about the for () command.

$$\begin{array}{l} > For(i \quad in \quad \circ : 5)\{ \\ > print(i) \\ > \} \\ [1] \quad \circ \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \\ > A = seq(1 : 3, by = 0.5) \\ > For(i \quad in \quad 2 : 7)\{ \\ > print(A[i] \quad - \quad A(i-1)] \\ > \} \\ [1]\ 0.5 \\ [1]\ 0.5 \\ [1]\ 0.5 \\ [1]\ 0.5 \\ [1]\ 0.5 \\ [1]\ 0.5 \\ [1]\ 0.5 \end{array}$$

2.14 While Command

Other statements for the conditional statements is while () command. In this command from the beginning, the amount of repetition is not clear.

In this case, as long as the logical expression in parentheses is correct, the loop continues. In the example below, the factorial function is written using the while statement command.

Example 2.34 Consider the following example of While () command.

$$
\begin{aligned}
&> factorial = Function(t)\{\\
&> F = 1\\
&> S = t\\
&> while(s > 1)\{\\
&> f = f * s\\
&> S = s - 1\\
&> return(f)\\
&> \}
\end{aligned}
$$

Example 2.35 Consider this example using the while statement.

$$
\begin{aligned}
&> i = 1\\
&> while(i<5)\{\\
&> \Pr\mathrm{int}(i)\\
&> i = i + 1\\
&> \}\\
&[1]\ 1\\
&[1]\ 2\\
&[1]\ 3\\
&[1]\ 4\\
&> X = 3\\
&> while(X<12)\{X = X + 2; if(X == 6)next; print(X);\}\\
&> \}\\
&[1]\quad 4\quad X\quad 8\quad 10
\end{aligned}
$$

The next command given above does not do anything. Only, if the logical condition of the condition is true it goes next X.

2.15 Repeat Command

Other functions for conditional commands is the function repeat (). In this function, the command inside the parenthesis is repeated until the logical expression inside the if statement is false. Finally, with the break command will leave the loop. Consider the factorial example that can be written with the repeat command.

Example 2.36 assign Consider the example of repeat command.

$$
\begin{aligned}
&> factorial = function\,(t)\,\{\\
&\quad > f = 1\\
&\quad > s = t\\
&\quad > repeat\{\\
&\quad > if(s<2)break\\
&\quad > f = f * s\\
&\quad > s = s - 1\\
&\quad > \}\\
&\quad > return(f)\\
&\quad > \}
\end{aligned}
$$

2.16 Import and Read Data in R

In R, there is a lot of data structures already available that can be used for training purposes and implementing models that users write.

2.16.1 Data Command

The data () command calls the specified file.

$$> data\,(hospital)$$

The result of this command is to call the hospital data.

We use the head () and tail () functions to show respectively the first and last 6 rows of the data.

$$> head(hospital)$$

If more than 6 rows are considered, their number should be mentioned.

$$> head(hospital, 8)$$

In this way, the first eight rows of data stored in the name of the hospital are called. Similarly, for the tail () command.

2.16.2 Scan Command

You can also use the scan () function to read the data.

This command is as follows.

$$> scan(file = \text{“ ”})$$

In front of the scan function, the filename must be mentioned with its address. Note that if the file structure is in the table, the scan command cannot read it.

The scan command feature is that it can read data from the keyboard.

Example 2.37 Reading data using scan () command.

```
> x = scan( )
1 : 15
2 : 18
3 : 20
4 : 5
5 :
Read 4 items
```

In the final step, pressing the Enter key can exit the data entry step.

Now we will call X.

```
> x
[1]  15  18  20  5
```

Note that if the data is in a string of characters, the scan function requires a $ argument.

Example 2.38

```
> scan (what = “  ”)
1 : a
2 : b
3 : c
4 :
read 3 items
[1]  “a”  “b”  “c”
```

2.16.3 Read.table Command

You can use the read.table () command to read data frames.

$$> Xdata = read.table(\text{“}C : /R/data.txt\text{”})$$

Notice that the “/” sign has been used in the path of the file. If the Windows operating system is used to specify the address, use the “\” sign.

For the read.table () command, the following arguments can be considered as mentioned in the following.

File:	This argument shows the name of the file which the data are to be read from. Each row of the table appears as one line of the file. If it does not contain an absolute path, getwd () should be used
Sep:	The character used to separate the values. If sep = “ ” then the values are separated by distance
Quote:	It shows the set of quoting characters which can be a single character string or NULL
Dec:	Character designated for decimal point
row.names:	The vector is the rows. If row.names = NULL then the digits are not executed for rows
Header:	The vector names of the columns are the default for this vector v
col.names:	The vector is the names of the columns
Nrows:	Maximum number of rows to be read (integer value)
Skip:	The number of rows that are not read from the first file (integer value)

Note read.table () command reads the .txt file.

If your file is xls or xlsx, to read such data you can save the file to *.txt format in Excel and read it with the read.table function. It can also be stored in *.csv format in Excel, so it calls it with the function below.

read.csv (“Name and address of file”)

2.16.4 Read.delim Command

You can also use the txt file to read the data from the following command.

$$read.delim(file, header = T, \ldots)$$

Arguments similar to those for the read.table command are also included in this command.

The default for this command for the distance between the different values is tab. But in read.table, you must specify the “sep” type and not the default.

If you need to retrieve a file multiple times in your program, you can do the following to simplify this. This way, save the address in a variable. You can then use it whenever you need it.

We do this by the following command.

$$> P = file.path(\text{“}FilenName.txt\text{”})$$

You can do this by calling the data () command.

$$> read.table(P, header = F, sep = \text{“.”})$$

Note that, “*header* = *F*” means that the first row of the data is not considered as the header. In the case that “*header* = *T*” the first row of the data is considered as the header.

2.16.5 Fread Command

In the case of high volume data (in a few million lines), the following command can be used:

$$\begin{aligned} &> X = Fread(\text{“}Filename, sep = \text{”}Auto\text{“},\\ &header = \text{“}Auto\text{”}, stringASFactors = F)\\ &> print(X) \end{aligned}$$

In this case, the result of the printed data is the first 5 rows and the last 5 lines of data. You can read a certain number of rows at the beginning of the file, if desired. This way we use nrows in the fread () command.

$$> P_1 = fread(\text{“}data.txt\text{”}, sep = \text{“}Auto\text{”}, header = \text{“}Auto\text{”}, select = 20)$$

Example 2.39 If we want to read a certain symbol of a row from a file, we will act as follows.

$$\begin{aligned} &> P_2 = Fread(\text{“}data.txt\text{”}, sep = \text{“}Auto\text{”},\\ &header = \text{“}Auto\text{”}, select = C(1, 3, 5)) \end{aligned}$$

Thus lines 1, 3 and 5 are read.

If the rows have names, you can use the select command as following.

$$select = C(\text{“}X_1\text{”}, \text{“}Y_1\text{”}, \text{“}Y_2\text{”})$$

2.16.6 Excel_sheets Command

If we want to insert pages in an excel file in R, we use the following command.

$$> s = excel\_sheets(\text{“}data.Xls\text{”})$$

The output of this command after executing the name of the page in the excel file is stored in sh1.

2.16.7 Read_excel Command

You can use the following command to read the values stored on each page of an excel file.

$$> read\_excel(\text{“}data.XLS\text{”}, sheet = \text{“}input\text{”})$$

After executing this command, the information stored in the page called “input” is read in the excel file and extracted from it.

2.17 Storage and Writing Data

The following commands can be used to write or save data.

2.17.1 Write.table Command

To write or save data to a file in txt or xls format, use the following command.

$$write.table(x, file = \text{“ ”}, \ldots)$$

That T is the data frame that is supposed to be stored. File= “filename” is the name of the file we want to store in x. Arguments of this command are similar to the read.table command. You can use the write.scv () command to store the data frames in the excel data format.

2.17.2 Sink Command

Also, the function sink () stores the output of a function or an application in a text file. In this command, we have to put the file name of the file you want to save in our sink () function.

$$> sink(\text{“}efficiency.txt\text{”})$$

By executing this command, the information stored in the txt file is stored in the name of efficiency.

2.18 Write Functions in R

The R environment allows users to save their frequently used functions in a new code so you can easily use it in its subsequent reps.

The general structure of a function is as follows.

$$\begin{aligned} &> Function(Arguments)\{ \\ &+ program \\ &+ \} \end{aligned}$$

Note that, the defined terms must be with “;” separate.

Consider the following example. This function takes the square from sum of the components of a vector.

Example 2.40 Consider the defined function () command.

$$\begin{aligned} &> X = C(2,3,5,6) \\ &> f = function(x)\{ \\ &+ sqrt(sum(x)) \\ &+ \} \\ &[1] \quad f = 4 \end{aligned}$$

If the program text contains only one command, you can refuse to write {}.

2.19 Convert Objects

In programming with R, it is possible to change the type of variables defined in a vector.

Example 2.41 Consider the following vector:

$$> x = (3, 1.2, \text{“}b\text{”})$$

With the class () command, we specify the vector type.

$$\begin{aligned} &> class(x) \\ &[1]\text{“}character\text{”} \\ &> x \\ &[1] \quad \text{“3”} \quad \text{“1.2”} \quad \text{“}b\text{”} \end{aligned}$$

As you can see, the vector x is a character type. Because in R if the vector contains characters, the whole vector is considered a type of character. As you can see, with calling the vector x, the position of its components is between two quotes, which is evidence that the entire vector is a type of character.

Example 2.42 Consider the following vector.

$$\begin{aligned} &> x = c(FALSE, 3, 2) \\ &> class(x) \\ &[1] \quad \text{“}numeric\text{”} \\ &> x \\ &[1] \quad 0 \quad 3 \quad 2 \end{aligned}$$

If the vector x is a numerical vector, then by calling the x-vector, instead of the logical expression FALSE, the numeric value is set to 0. As is clear, the vector y is a numerical vector. In R, if the vector contains TRUE and FALSE logical numbers and phrases, it considers the type of vector as numeric.

Example 2.43 Consider the following vector.

$$> x = c(\text{“}b\text{”}, TRUE, \text{“}a\text{”})$$

This vector is of character type.

$$\begin{aligned} &> class(x) \\ &[1] \quad \text{“}character\text{”} \\ &> x \\ &[1] \quad \text{“}b\text{”} \quad \text{“}TRUE\text{”} \quad \text{“}a\text{”} \end{aligned}$$

All elements of this vector are considered as characters.

Thus, if the vector contains logical characters and phrases, then this vector is considered as the character type by default.

In R, it is not possible to define vectors of the combination of elements, and R itself presupposes a special type for these compound vectors as described in the example above.

Consider the vector below. In this section, we introduce a function by which the user can convert vector types to each other.

Example 2.44 Consider y vector and it class.

```
> y = c(1 : 10)
> class(y)
[1]  "integer"
> y
 > [1]  1  2  3  4  5  6  7  8  9  10
```

As is clear, the type of vector y that is defined above is correct. To convert this vector type correctly, use the following command. In this way, the numeric class of the y vector of the correct type is obtained as follows.

Example 2.45 Consider the as.numeric () command.

```
> y = as.numeric(x)
> class(y)
[1]  "numeric"
> 7
 [1]  1  2  3  4  5  6  7  8  9  10
```

Note that with this statement, the class of vector of the intended variable is changed, not the numbers themselves. As can be seen, the same result is obtained by calling the vectors y and z.

Now if we convert the integer class of vector y changed to logical then we have:

```
> w = as.logical(y)
```

In this change, the number 0 corresponds to the logical FALSE expression and the remainder of the numbers corresponding to the TRUE logical expression.

```
> w
[1]  F  T  T  T  T  T  T  T  T  T
```

In R, the letters T and F can be used to replace TRUE and FALSE.

```
> class(w)
[1]  "logical"
```

To get the class of vector *y*, one should use the following command.

Example 2.45 Consider the as.charachter () command.

$$
\begin{aligned}
&> p = as.character(x)\\
&> class(p)\\
&[1] \quad \text{“}character\text{”}\\
&> p\\
&[1] \quad \text{“1”} \quad \text{“2”} \quad \text{“3”} \quad \text{“4”} \quad \text{“5”} \quad \text{“6”} \quad \text{“7”} \quad \text{“8”} \quad \text{“9”} \quad \text{“10”}
\end{aligned}
$$

Note this conversion is not possible in the other way.

Example 2.46 For example, consider the character y vector.

$$> y = c(\text{“}a\text{”}, \text{“}b\text{”})$$

The class of this vector is not numeric.

$$
\begin{aligned}
&> 7 = as.numeric(y)\\
&[1] \quad NA \quad NA \quad NA \quad (warning \quad message: \quad NAS \quad \text{int}raduced \quad by \quad coercion)
\end{aligned}
$$

By executing this command, R declares your coding error. Because in executing this command, there was no amount for the result of this conversion (which also comes to mind).

Consider the following matrix to avoid the above error, you should use the following command.

Example 2.48 Consider the as.matrix () command.

$$
\begin{aligned}
&> y = as.matrix(x)\\
&> \dim(x)\\
&[1] \quad 6 \quad 1\\
&> nrow(y)\\
&[1] \quad 1\\
&> ncol(y)\\
&[1] \quad 6\\
&>
\end{aligned}
$$

Example 2.49 Consider the following data frame.

$$
\begin{aligned}
&> school\\
&\qquad\qquad ST.1 \quad ST.2\\
&[1,] \qquad 17 \quad 11\\
&[2,] \qquad 13.25 \quad 15\\
&[3,] \qquad 12 \quad 17\\
&[4,] \qquad 11 \quad 14.5\\
&[5,] \qquad 20 \quad 11.75\\
&> length(school)\\
&[1] \quad 2
\end{aligned}
$$

This data framework can be considered as a matrix.

Example 2.50 Consider the is.data.frame (), is.list (), and is.matrix () commands.

```
> as.matrix(school)
      class6     class2
[1,]  "17"       "11"
[2,]  "13.25"    "15"
[3,]  "12"       "17"
[4,]  "11"       "14.5"
[5,]  "20"       "11.75"
> y = as.matrix(school)
> dim(y)
[1]  5  2
> length(y)
> 10
> is.matrix(school)
[1]  FALSE
> is.data.frame(school)
[1]TRUE
> is.list(school)
[1]  TRUE
```

Example 2.51 Consider the vector x as defined below.

```
> x
[1]  ∘  1  2  3  4
```

Now let's define *y* as a logical vector.

```
> y = as.logical(x)
> y
[1]  F  T  T  T
```

Note that, F corresponds to the number "0" and the T is referring to any nonzero value.

```
> class(y)
[1]  "logical"
```

Now, assume

```
> x = c("a", "b")
> as.numeric(x)
[1]  NA  NA
  warning message
```

Table 2.2 Rules for vector conversion

Rules	Used function	Vector conversion
$FALSE \quad , \quad F \rightarrow 0$ $TRUE \quad , \quad T \rightarrow 1$ "1", "2", ... $\rightarrow 1, 2, \ldots$ "a", ... $\rightarrow NA$	*as.numeric*	*To numeric class*
$F \rightarrow 0$ *Other numbers* $\rightarrow T$ "*FALSE*", "*F*" $\rightarrow$ *FALSE* "*TRUE*", "*T*" $\rightarrow$ *TRUE* *other characters* $\rightarrow$ *NA*	*As.logical*	*To logical class*
$1, 2, \ldots \rightarrow$ "1", "2", ... $FALSE, F \rightarrow$ "*FALSE*" $TRUE, T \rightarrow$ "*TRUE*"	*As.character*	*To character class*

A logical error occurred because the character cannot be converted to numeric value.

Similarly

$$> as.logical(x)$$
$$[1] \quad NA \quad NA$$

Because the character cannot be converted to logical value. Note that "NA" means "Not Available".

Consider the following Table 2.2. In these tables, we summarize the conversion of a variety of vector classes in R.

2.20 Conclusion

In this chapter we have provided some basic definitions of R and required commands used in writing DEA models with R. We have reviewed all the commands sequentially according to their concepts and provided numerical examples for each command in order to clarify the usage of commands for the readers. Now, the readers should have the ability to identify the various objects that can be used in R. Also, they are aware of how to use mathematical functions in R, and use R for calculations. Readers have been introduced to conditional structures, as well as how to write a function, reading data from a file and writing data on a file.

References

1. T.M. Davies, *The Book of R*. (No Starch Press, California, 2016), p. 832. ISBN-13:978-1-59327-651-5
2. N. Matloff, *The Art of R Programming*. (Publisher William Pollock, California, 2011). ISBN-13: 978-1-59327-384-2
3. R. Cotton, *Learning R*. (OReilly Media, California, 2013). ISBN: 9781449357160
4. R. Kabacoff. *R in action*. (Manning Publication Co, New York, 2015), 608p. ISBN 9781617291388
5. N. Zumel, J. Mount, *Practical Data Science with R*. (Manning Publication Co, 2014), 416p. ISBN 9781617291562
6. J.P. Lander, *R for Everyone: Advanced Analytics and Graphics*. (Addison-Wesley Professional, Boston, 2014). ISBN 9780321888037
7. A. de Vries, J. Meys, *R for Dummies*. (Wiley, Hoboken, 2015). ISBN: 978-1-119-96284-7
8. F. Alvaro, *Easy R Programming for Beginners*. (Create Space Independent Publishing Platform, California, 2016). ISBN 13: 9781533685018
9. J. Adler, *R in a Nutshell*. (OReilly Media, 2012), 72413p. ISBN 9781449312084
10. C. Kleiber, A. Zeileis, *Applied Econometrics with R (Use R!)*. (Springer, 2008). ISBN 13: 978-0387773162
11. G. Petris, S. Petrone, *Dynamic Linear Models with R (Use R!)*. (Springer, 2009). ISBN 13: 978-0387772370
12. P.S.P. Cowpertwait, A.V. Metcalfe, *Introductory Time Series with R (Use R!)*. (Springer, 2009). ISBN 13: 978-0387886978
13. B. Everitt, T. Hothorn, *An Introduction to Applied Multivariate Analysis with R (Use R!)*. (Springer, 2011). ISBN 978-1-4419-9650-3
14. W.E. Hart, C. Laird, J. Watson, D.L. Woodruff, *Pyomo—Optimization Modeling in Python* (Springer, Berlin, 2012) ISBN 978-1-4614-3226-5. ISBN 978-3-319-58821-6
15. J. Ledolter, *Data Mining and Business Analytics with R*. (Wiley, 2013). ISBN 13: 978-1118447147
16. K.M. Ramachandran, C.P. Tsokos, *Mathematical Statistics with Applications in R*. (Academic Press, 2014). ISBN 13: 978-0124171138
17. V.A. Bloomfield, *Using R for Numerical Analysis in Science and Engineering*. (Chapman & Hall/CRC The R Series, 2014). ISBN-13: 978-0124171138
18. M.J. *Crawley, Statistics: An Introduction Using R*. (Wiley, 2014). ISBN-13: 978-1118941096
19. C. Paulo, *Modern Optimization with R*. (Springer, 2014). ISBN 978-3-319-08263-9
20. C.L. Emilio, M.M. Javier, P.C. Mariano, *Quality Control with R, An ISO Standards Approach*. (Springer, 2015). ISBN 978-3-319-24046-6
21. J.L. Taveras, *R for Excel Users: An Introduction to R for Excel Analysts*. (Create Space Independent Publishing Platform, 2016). ISBN-13: 978-1500566357
22. B. Pfaff, *Financial Risk Modelling and Portfolio Optimization with R*. (Wiley, 2016). ISBN: 978-1-119-11966-1
23. G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*. (Springer, 2017). ISBN-13: 978-1461471370. (Springer Texts in Statistics)
24. R. Yao, *R: In 8 Hours, For Beginners, Learn Coding Fast!*. (Independently Published, 2018). ISBN-13: 978-1731132178
25. R Core Team, *R Language Definition. version 3.5.2 (2018-12-20)*. (2018). https://cran.r-project.org
26. R Core Team, *R Data Import/Export. version 3.5.2 (2018-12-20)*. https://cran.r-project.org

27. R Core Team, *An Introduction to R. Notes on R: A Programming Environment for Data Analysis and Graphics. Version 3.5.2 (2018-12-20)*. https://cran.r-project.org/
28. R Core Team, *Internals. Version 3.5.2 (2018-12-20)*. (2018). https://cran.r-project.org
29. R Core Team, *A Language and Environment for Statistical Computing. Reference Index. Version 3.5.2 (2018-12-20)*. (2018). https://cran.r-project.org/
30. R Core Team, *Writing R Extensions. Version 3.5.2 (2018-12-20)*. (2018). https://cran.r-project.org/

Chapter 3
Basic DEA Models with R Codes

Abstract As a powerful assessment tool, DEA is very famous in variety of areas. Different applications are performed using EA models for relative evaluations. Being familiar with DEA models and their characteristics is necessary as a mathematical tool for evaluation. In this chapter at first some basic models of DEA technique are briefly reviewed and introduced literally and mathematically. Then, R codes of these models along with numerical examples are provided for readers.

Keywords DEA models · Efficiency scores · Benchmark units · Efficient and inefficient units

3.1 Introduction

Data Envelopment Analysis (DEA) is a mathematical programming approach for evaluating the batch performance of decision making units. DEA provides facilities for studying and evaluating units with multiple inputs and multiple outputs. It is based on linear algebra. It can be said that the capabilities of this technique are mostly due to the use of linear programming. In fact, DEA is able to use linear programming methods and duality theorems to determine the source and amount of inefficiency for each input and output. DEA also makes use of this technique for many opportunities for collaboration between analysts and decision-makers. These collaborations can be used to select the inputs and outputs of the units under evaluation and how the decision-making units are act and target the inefficient units. This technique was introduced by Farrell in 1957 and then developed by Charnes et al. [1]. In this chapter, we introduce the basic models of efficiency estimation using DEA technique and then present the corresponding R codes for these models.

F. Hosseinzadeh Lotfi et al., *Data Envelopment Analysis with R*, Studies in Fuzziness and Soft Computing 386, https://doi.org/10.1007/978-3-030-24277-0_3

3.2 Input-Oriented DEA Models with R Codes

In this section, R codes for input-oriented DEA envelopment and multiplier models are presented.

3.2.1 Input-Oriented CCR Envelopment Model with R Code

Input-oriented CCR envelopment model has been presented by Charnes et al. [1]. The optimal value of this model shows the relative efficiency of the unit under evaluation. The vector and matrix forms of this model are as follows:

$$\begin{aligned} \min \quad & \theta \\ \text{s.t.} \quad & -\theta x_{ip} + \sum_{j=1}^{m} \lambda_i x_{ij} \le 0, \quad i = 1, \ldots, m, \\ & \sum_{j=1}^{n} \lambda_i x_{rj} \ge y_{rp}, \quad r = 1, \ldots, s, \\ & \lambda_j \ge 0, \quad j = 1, \ldots, n, \end{aligned} \tag{3.1}$$

$$\begin{aligned} \min \quad & [1, 0, ..., 0][\theta, \lambda_1, ..., \lambda_n]^t \\ s.t. \quad & \left[\begin{array}{c|ccc} -x_{1p} & x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & & \vdots \\ -x_{mp} & x_{m1} & \cdots & x_{mn} \\ \hline 0 & y_{11} & \cdots & y_{1n} \\ \vdots & \vdots & & \vdots \\ 0 & y_{s1} & \cdots & y_{sn} \end{array}\right] \left[\begin{array}{c} \theta \\ \hline \lambda_1 \\ \vdots \\ \lambda_n \end{array}\right] \begin{array}{c} \le_m \\ \hline \ge_s \end{array} \left[\begin{array}{c} 0_{m\times 1} \\ \hline y_{1p} \\ \vdots \\ y_{sp} \end{array}\right] \\ & \lambda_j \ge 0, \qquad j = 1, .., n. \end{aligned} \tag{3.2}$$

In matrix form model (3.2), the subscripts *"m"* and "*s*" in $\le_m$ and $\ge_s$ indicate the number of constraints with sings $\le$ and $\ge$, respectively. Also, need to be noted that all variables are by default considered as non-negative in corresponding R codes because of using the LP () command in these codes. Regarding what version of the R software is used, it may be necessary to use different libraries to use the write command. It should also be noted that the write.csv ( ) command can easily be used for writing

The R code of model (3.2) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
```

```
# reading data
df = data.frame(read_excel(path = "data1.xlsx", sheet="1"))
# defining a matrix to be filled with input and output data
f.con=matrix(ncol=N+1,nrow=m+s)
# inputs and outputs to be used for deriving targets
inputs=data.frame(df[1:2])
outputs=data.frame(df[3:4])
# number of elements
m=2
s = ncol(df)-m
N = nrow(df)
for (j in 1:N) {
# defining right handside, directions, and objective
f.rhs = c(rep(0,m),df[j,(m+1):(m+s)])
f.dir = c(rep("<=",m),rep(">=",s))
f.obj = c(1, rep(0,N))
#defining matrix of coefficient
for(i in 1:m){
f.con[i,1:(N+1)] = c(as.numeric(-df[j,i]),df[,i])
}
for(r in (m+1):(s+m)) {
f.con[r,1:(N+1)] = c(0,as.numeric(df[,r]))
}
# solving the model
results = lp('min', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weights = results$solution[1]
lambdas = results$solution[seq(2,N+1)]
xbench = lambdas %*% as.matrix(inputs)
ybench = lambdas %*% as.matrix(outputs)
} else {
weights = rbind(weights, results$solution[1])
lambdas = rbind(lambdas, results$solution[seq(2,N+1)])
xbench = lambdas %*% as.matrix(inputs)
ybench = lambdas %*% as.matrix(outputs)
}
}
Es = data.frame(weights, lambdas,xbench, ybench)
colnames(Es) = c('effi', rep("lambda",N),rep("x benchmark", m), rep("y
benchmark", s))
WriteXLS(Es,"1-CCR-ENV_INP.xls", row.names = FALSE, col.names =
TRUE)
```

Table 3.1 Data1 (Input and output data)

DMU	I_1	I_2	O_1	O_2
1	20	11	8	30
2	11	40	21	20
3	32	30	34	40
4	21	30	18	50
5	20	11	6	17
6	12	43	23	58
7	7	45	28	30
8	31	45	40	20
9	19	22	27	23
10	32	11	38	45

Example 3.1 Consider the data set as introduced in Table 3.1 with two inputs and two outputs for ten DMUs. The results of running this code are listed in Tables 3.2 and 3.3. These results show the efficiency score (EFFI) for each DMU, as well as optimal values of intensifier variables (Lan*) and target of inputs (Tx) and outputs (Ty). Note that benchmark units have been obtained according to $\left(\sum_{j=1}^{n} \lambda_j^* x_{ij}, \sum_{j=1}^{n} \lambda_j^* y_{ij}\right)$.

3.2.2 *Input-Oriented CCR Multiplier Model with R Code*

Input-oriented CCR multiplier model has been introduced by Charnes et al. [1]. The optimal objective value shows the relative efficiency of the (DMU) unit under evaluation. Here u and v show the input and output weights. According to the duality theorem the finite optimal objective function value of the primal and dual (Input-oriented CCR enveloping model) problems are the same. Vector and matrix forms of this model are as follows.

$$\begin{aligned} \max \quad & \sum_{r=1}^{s} u_r y_{rP} \\ \text{s.t.} \quad & -\sum_{i=1}^{m} v_i x_{ij} + \sum_{r=1}^{s} u_r y_{rj} \le 0, \quad j = 1, \ldots, n, \\ & \sum_{i=1}^{m} v_i x_{ip} = 1, \\ & u_r \ge 0, \qquad v_i \ge 0, \qquad r = 1, \ldots, s, \quad i = 1, \ldots, m. \end{aligned} \tag{3.3}$$

Table 3.2 Results of model (3.1), λ^* and target points

DMU	EFFI.	Lan_1*	Lan_2*	Lan_3*	Lan_4*	Lan_5*	Lan_6*	Lan_7*	Lan_8*	Lan_9*	Lan_{10}*
1	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.762	0.792	0.000	0.000	0.000	0.000	0.000	0.000	0.085	0.000	0.000
3	0.961	1.420	0.000	0.000	0.000	0.000	0.000	0.000	0.113	0.000	0.000
4	0.625	0.833	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	1.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
6	0.440	0.361	0.000	0.000	0.000	0.090	0.000	0.000	0.000	0.000	0.000
7	0.893	0.833	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
9	0.815	0.914	0.000	0.000	0.000	0.371	0.000	0.000	0.000	0.000	0.000
10	0.202	0.679	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 3.3 Results of model (3.1), λ^* and target points

DMU	Tx_1	Tx_2	Ty_1	Ty_2
1	15.000	27.000	45.000	78.000
2	18.277	24.370	44.000	68.260
3	29.802	42.300	75.000	119.352
4	12.500	22.500	37.500	65.000
5	18.000	83.000	86.000	76.000
6	7.036	17.236	24.000	35.000
7	12.500	22.500	37.500	65.000
8	75.000	35.000	98.000	76.000
9	20.381	55.436	73.000	99.458
10	10.192	18.346	30.577	53.000

$$\begin{aligned}
&\max\left[0,\ldots,0,y_{1p},\ldots,y_{sp}\right]\left[v_1,\ldots,v_m,u_1,\ldots,u_s\right]^t\\
&s.t.\left[\begin{array}{ccc|ccc}
-x_{11} & \cdots & -x_{m1} & y_{11} & \cdots & y_{s1}\\
\vdots & & \vdots & \vdots & & \vdots\\
-x_{1n} & \cdots & -x_{mn} & y_{1n} & \cdots & y_{sn}\\
\hline
x_{1p} & \cdots & x_{mp} & 0 & \cdots & 0
\end{array}\right]\left[\begin{array}{c}
v_1\\ \vdots\\ v_m\\ \hline u_1\\ \vdots\\ u_s
\end{array}\right]\begin{array}{l}\leq_n\\ =_1\end{array}\left[\begin{array}{c}0_{n\times 1}\\ \hline 1\end{array}\right]\\
&u_r\geq 0,\quad v_i\geq 0,\quad r=1,\ldots,s,\ \ i=1,\ldots,m.
\end{aligned}\tag{3.4}$$

The R code of model (3.4) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df = data.frame(read_excel(path = "data1.xlsx", sheet="1"))
# number of elements
m=2
s = ncol(df)-m
N = nrow(df)
#defining right hand side, and directions
f.rhs = c(rep(0,N),1)
f.dir = c(rep("<=",N),"=")
for (j in 1:N) {
#defining objective, and matrix of coefficient
f.obj = c(rep(0,m),df[j,(m+1):(m+s)])
```

```
con1 = cbind(-df[,1:m],df[,(m+1):(m+s)])
con2 = c(as.numeric(df[j,1:m]), rep(0,s))
f.con = rbind(con1, con2)
#solving model
results = lp ("max",f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=TRUE)
if (j==1) {
weights = results$solution
effcrs = results$objval
lambdas = results$duals[seq(1,N)]
} else {
weights = rbind(weights, results$solution)
effcrs = rbind(effcrs, results$objval)
lambdas = rbind(lambdas, results$duals[seq(1,N)]) }
}
ls = data.frame(effcrs, weights,lambdas)
colnames(ls) = c('effi', rep("v",m), rep("u", s), rep("lambda", N))
WriteXLS(ls,"2-CCR-MUL-INP.xls", row.names = F, col.names = T)
```

Example 3.2 Consider the data set given in Table 3.1. The efficiency scores (EFFI) as well as optimal intensifier variables (Lan*) and optimal input and output weights (v*, u*) are obtained from solving model (3.3) are gathered in Tables 3.4 and 3.5.

Note that in this code "compute.sens" is set to be T (true), thus dual variables can be obtained from the command of LP. In this way, "results$duals[seq(1,N)]" will results the optimal values of intensifier variable $\left(\lambda_1^*, \ldots, \lambda_n^*\right)$ without solving the enveloping form of the model (3.3) directly.

Table 3.4 Optimal solution of model (3.4)

DMU	Lan_1*	Lan_2*	Lan_3*	Lan_4*	Lan_5*	Lan_6*	Lan_7*	Lan_8*	Lan_9*	Lan_{10}*
1	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.792	0.000	0.000	0.000	0.000	0.000	0.000	0.085	0.000	0.000
3	1.420	0.000	0.000	0.000	0.000	0.000	0.000	0.113	0.000	0.000
4	0.833	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
6	0.361	0.000	0.000	0.000	0.090	0.000	0.000	0.000	0.000	0.000
7	0.833	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
9	0.914	0.000	0.000	0.000	0.371	0.000	0.000	0.000	0.000	0.000
10	0.679	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 3.5 Efficiency scores and optimal weights

DMU	EFFI.	v_1*	v_2*	u_1*	u_2*
1	1.000	0.000	0.037	0.000	0.013
2	0.762	0.012	0.022	0.017	0.000
3	0.961	0.009	0.016	0.013	0.000
4	0.625	0.050	0.000	0.000	0.010
5	1.000	0.022	0.007	0.012	0.000
6	0.440	0.063	0.000	0.005	0.009
7	0.893	0.071	0.000	0.000	0.014
8	1.000	0.007	0.013	0.010	0.000
9	0.815	0.021	0.007	0.011	0.000
10	0.202	0.000	0.011	0.000	0.004

Consider the obtained results of model (3.4) listed in Tables 3.4 and 3.5. As models (3.1) and (3.3) are primal and dual models, thus they have the same efficiency scores.

3.2.3 *Input-Oriented BCC Multiplier Model with R Code*

Input-oriented BCC multiplier model has been introduced by Banker et al. [2]. The optimal objective function value shows the relative efficiency of the DMU under evaluation (DMU_p). Here u_0 is a variable free in sign which is considered as a subtraction of two positive variables u_0^+ and u_0^-. Vector and matrix forms of this model are as follows.

$$\begin{aligned}
\max \quad & \sum_{r=1}^{s} u_r y_{rP} + u_0^+ - u_0^- \\
\text{s.t.} \quad & -\sum_{i=1}^{m} v_i x_{ij} + \sum_{r=1}^{s} u_r y_{rj} + u_0^+ - u_0^- \le 0, \quad j = 1, \ldots, n, \\
& \sum_{i=1}^{m} v_i x_{ip} = 1, \\
& u_r \ge 0, \quad v_i \ge 0, \quad r = 1, \ldots, s, \quad i = 1, \ldots, m, \\
& u_0^+ \ge 0, \quad u_0^- \ge 0.
\end{aligned} \tag{3.5}$$

$$\max\left[0,...,0,y_{1p},...,y_{sp},1,-1\right]\left[v_1,...,v_m,u_1,...,u_s,u_0^+,u_0^-\right]^t$$

$$s.t.\quad \left[\begin{array}{ccc|ccc|cc} -x_{11} & \cdots & -x_{m1} & y_{11} & \cdots & y_{s1} & 1 & -1 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots \\ -x_{1n} & \cdots & -x_{mn} & y_{1n} & \cdots & y_{sn} & 1 & -1 \\ \hline x_{1p} & \cdots & x_{mp} & 0 & \cdots & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hline u_1 \\ \vdots \\ u_s \\ \hline u_0^+ \\ u_0^+ \end{array}\right] \begin{matrix} \leq_n \\ =_1 \end{matrix} \left[\begin{array}{c} 0_{n\times 1} \\ \hline 1 \end{array}\right] \tag{3.6}$$

$$u_r \geq 0, \quad v_i \geq 0, \qquad r=1,...,s, \quad i=1,...,m,$$
$$u_0^+ \geq 0, \quad u_0^- \geq 0.$$

The R code of model (3.6) is as follows.

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
# defining a matrix to be filled with input and output data
df = data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# defining inputs and outputs to be used for deriving targets
inputs=data.frame(df[1:2])
outputs=data.frame(df[3:4])
#number of elements
m=2
s = ncol(df)-m
N = nrow(df)
#defining right hand side, directions
f.rhs = c(rep(0,N),1)
f.dir = c(rep("<=",N),"=")
for (j in 1:N) {
#defining objective, and matrix of coefficient
f.obj = c(rep(0,m),as.numeric(df[j,(m+1):(m+s)]), 1, -1)
con1 = cbind(-df[,1:m],df[,(m+1):(m+s)], 1, -1)
con2= c(as.numeric(df[j,1:m]), rep(0,s),0,0)
```

```
f.con = rbind(con1,con2)
#solving model
resultss = lp ("max",f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=TRUE)
multiplierss = resultss$solution
u0 = multiplierss[s+m+1]-multiplierss[s+m+2]
if (j==1) {
weightss = c(multiplierss[seq(1,s+m)],u0)
effvrss = resultss$objval
lambdass= resultss$duals[seq(1,N+1)]
xbench = lambdas %*% as.matrix(inputs)
ybench = lambdas %*% as.matrix(outputs)
} else {
weightss = rbind(weightss,c(multiplierss[seq(1,s+m)],u0))
effvrss = rbind(effvrss, resultss$objval)
lambdass = rbind(lambdass,resultss$duals[seq(1,N+1)])
xbench = lambdas %*% as.matrix(inputs)
ybench = lambdas %*% as.matrix(outputs) }
}
ls = data.frame(effvrss, weightss, lambdass, xbench, ybench)
colnames(ls)=c('effi', rep('v', m), rep('u', s), 'u0', rep("lambda", N), "teta",
rep("xbenchmark", m), rep("y benchmark", s))
WiteXLS(ls,"3-BCC-MUL-INP.xls", row.names = F, col.names = T)
```

Example 3.3 Consider the data set given in Table 3.1. The results of model (3.5) are listed in Table 3.6. Also, the optimal input and output weights are listed in Table 3.7. Moreover, as "compute.sens" is equal to "TRUE" thus it is possible to have the optimal values of $\left(\lambda_1^*, \ldots, \lambda_n^*\right)$ without solving correspondence enveloping

Table 3.6 Results of model (3.5)

DMU	EFFI.	v_1*	v_2*	u_1*	u_2*	u_0*
1	1.000	0.000	0.037	0.000	0.013	0.000
2	0.844	0.000	0.031	0.000	0.000	0.844
3	1.000	0.011	0.015	0.014	0.000	−0.072
4	0.731	0.050	0.000	0.002	0.000	0.643
5	1.000	0.022	0.007	0.012	0.000	0.000
6	0.875	0.063	0.000	0.003	0.000	0.804
7	1.000	0.071	0.000	0.000	0.005	0.643
8	1.000	0.007	0.013	0.010	0.000	0.000
9	0.851	0.016	0.009	0.013	0.000	−0.127
10	0.297	0.000	0.011	0.000	0.000	0.297

Table 3.7 Optimal values of intensifier variable

DMU	Lan_1*	Lan_2*	Lan_3*	Lan_4*	Lan_5*	Lan_6*	Lan_7*	Lan_8*	Lan_9*	Lan_{10}*
1	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.619	0.000	0.000	0.000	0.000	0.000	0.381	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
9	0.235	0.000	0.307	0.000	0.458	0.000	0.000	0.000	0.000	0.000
10	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 3.8 Efficiency scores and target points

DMU	θ*	Tx_1	Tx_2	Ty_1	Ty_2
1	1.000	15.000	27.000	45.000	78.000
2	0.844	18.277	24.370	44.000	68.260
3	1.000	29.802	42.300	75.000	119.352
4	0.731	12.500	22.500	37.500	65.000
5	1.000	18.000	83.000	86.000	76.000
6	0.875	7.036	17.236	24.000	35.000
7	1.000	12.500	22.500	37.500	65.000
8	1.000	75.000	35.000	98.000	76.000
9	0.851	20.381	55.436	73.000	99.458
10	0.297	10.192	18.346	30.577	53.000

model directly. Pay attention that in LP command compute.sens is set to TRUE thus considering resultss$duals[seq (1, N+1)] will indicate dual variables, λ in enveloping form of model, and then target units are computed by $\left(\sum_{j=1}^{n} \lambda_j^* x_{ij}, \sum_{j=1}^{n} \lambda_j^* y_{ij}\right)$. The target points are listed in Table 3.8. The symbol θ^* in Table 3.8 obtained from dual variables is the same score as EFFI in Table 3.6.

Consider this note that the free variable u0* is obtained from *"*u0 = multipliers[s +m+1]-multipliersss[s+m+2]*"*.

3.2.4 Input-Oriented BCC Envelopment Model with R Code

Input-oriented BCC envelopment model has been introduced by Banker et al. [2]. The optimal objective function value shows the relative efficiency of the DMU under evaluation (DMU_p). According to the duality theorem the finite optimal objective function values of enveloping and multiplier BCC models are same as

they are a pair of primal and dual problems. The vector and matrix forms of this model are as follows.

$$\begin{aligned} \min \quad & \theta \\ \text{s.t.} \quad & -\theta x_{ip} + \sum_{i=1}^{u} \lambda_i x_{ij} \le 0, \quad j = 1, \ldots, m, \\ & \sum_{i=1}^{u} \lambda_i y_{rj} \ge y_{rp}, \qquad r = 1, \ldots, s, \\ & \sum_{i=1}^{u} \lambda_i = 1, \\ & \lambda_j \ge 0, \qquad j = 1, \ldots, n. \end{aligned} \tag{3.7}$$

$$\begin{aligned} &\min \quad [1, 0, ..., 0][\theta, \lambda_1, ..., \lambda_n]^t \\ &s.t. \left[\begin{array}{c|ccc} -x_{1p} & x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & & \vdots \\ -x_{mp} & x_{m1} & \cdots & x_{mn} \\ \hline 0 & y_{11} & \cdots & y_{1n} \\ \vdots & \vdots & & \vdots \\ 0 & s_{s1} & \cdots & y_{sn} \\ 0 & 1 & \cdots & 1 \end{array}\right] \left[\begin{array}{c} \theta \\ \hline \lambda_1 \\ \vdots \\ \lambda_n \end{array}\right] \begin{array}{c} \le_m \\ \ge_s \\ =_1 \end{array} \left[\begin{array}{c} 0_{m\times 1} \\ \hline y_{1p} \\ \vdots \\ y_{sp} \\ \hline 1 \end{array}\right] \\ &\lambda_j \ge 0, j = 1, ..., n. \end{aligned} \tag{3.8}$$

The R code of model (3.8) is as follows.

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
library(xlsxjars)
#reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet="1"))
# defining inputs and outputs to be used for deriving targets
inputs=data.frame(df[1:2])
outputs=data.frame(df[3:4])
#number of elements
m=2
```

```
s=ncol(df)-m
N = nrow(df);N
# defining a matrix to be filled with input and output data
f.con=matrix(ncol=N+1,nrow=m+s+1)
for (j in 1:N) {
#defining right handside, directions, and objective
f.rhs = c(rep(0,m),unlist(unname(df[j,(m+1):(m+s)])), 1)
f.dir = c(rep("<=",m),rep(">=",s), "=")
f.obj = c(1, rep(0,N))
#defining matrix of coefficient
for(i in 1:m) {
f.con[i,1:(N+1)]=c(-df[j,i],df[,i]) }
for(r in (m+1):(s+m)) {
f.con[r,1:(N+1)]=c(0,df[,r]) }
f.con[m+s+1, 1:(N+1)]=c(0, rep(1,N))
#solving model
results = lp('min', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
eff = results$objval
lambdas = results$solution[2:(N+1)]
xbench = lambdas %*% as.matrix(inputs)
ybench = lambdas %*% as.matrix(outputs)
} else {
eff = rbind(eff, results$objval)
lambdas = rbind(lambdas, results$solution[2:(N+1)])
xbench = lambdas %*% as.matrix(inputs)
ybench = lambdas %*% as.matrix(outputs) }
}
Es = data.frame(eff, lambdas, xbench, ybench)
colnames(Es) = c('effi', rep('lambda', N), rep("x benchmark", m), rep("y
benchmark", s))
WriteXLS(Es,"4-BCC-ENV-INP.xls", row.names=F, col.names=T)
```

Example 3.4 Consider the data set given in Table 3.1 with two inputs and two outputs for ten DMUs. The obtained results show the efficiency scores (EFFI) as well as optimal values of intensifier variables (Lan*), and input and output targets, (Tx*) and (Ty*). The obtained results are listed in Table 3.9 and Table 3.10.

Table 3.9 Efficiency scores and optimal values of intensifier variable of model (3.6)

DMU	EFFI	Lan_1*	Lan_2*	Lan_3*	Lan_4*	Lan_5*	Lan_6*	Lan_7*	Lan_8*	Lan_9*	Lan_{10}*
1	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.844	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	1.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.731	0.619	0.000	0.000	0.000	0.000	0.000	0.381	0.000	0.000	0.000
5	1.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
6	0.875	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
7	1.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
8	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
9	0.851	0.235	0.000	0.307	0.000	0.458	0.000	0.000	0.000	0.000	0.000
10	0.297	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 3.10 Input and output of target points

DMU	Tx_1	Tx_2	Ty_1	Ty_2
1	15.000	27.000	45.000	78.000
2	15.000	27.000	45.000	78.000
3	31.000	44.000	75.000	24.000
4	14.619	38.048	37.000	73.048
5	18.000	83.000	86.000	76.000
6	14.000	56.000	24.000	65.000
7	14.000	56.000	24.000	65.000
8	75.000	35.000	98.000	76.000
9	21.283	57.890	73.000	60.520
10	15.000	27.000	45.000	78.000

3.3 Output-Oriented DEA Models with R Codes

In this section, R codes for output-oriented DEA envelopment and multiplier models are presented.

3.3.1 Output-Oriented CCR Envelopment Model with R Code

Output-oriented CCR envelopment model has been introduced by Charnes et al. [1]. The optimal value shows the relative efficiency of the DMU under evaluation (DMU_p). In accordance to the strong duality theorem having finite optimal value, this model and the previous model (Output-oriented CCR multiplier model) have same values. Vector and matrix forms of this model can be stated as follows.

$$\begin{aligned} \max \quad & \varphi \\ \text{s.t.} \quad & \sum_{j=1}^{n} \lambda_j x_{ij} \le x_{ip}, && i = 1, \ldots, m, \\ & -\varphi y_{rp} + \sum_{j=1}^{n} \lambda_i y_{rj} \ge 0, && r = 1, \ldots, s, \\ & \lambda_j \ge 0, && j = 1, \ldots, n. \end{aligned} \tag{3.9}$$

$$\begin{aligned}
&\max \quad [1,0,...,0][\varphi,\lambda_1,...,\lambda_n]^t \\
&s.t. \begin{bmatrix} 0 & x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & & \vdots \\ 0 & x_{m1} & \cdots & x_{mn} \\ \hline -y_{ip} & y_{11} & \cdots & y_{1n} \\ \vdots & \vdots & & \vdots \\ -y_{sp} & s_{s1} & \cdots & y_{sn} \end{bmatrix} \begin{bmatrix} \varphi \\ \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix} \begin{matrix} \leq_m \\ \geq_s \end{matrix} \begin{bmatrix} x_{1p} \\ \vdots \\ x_{mp} \\ \hline 0_{s\times 1} \end{bmatrix} \\
&\lambda_j \geq 0, \qquad\qquad j = 1,...,n.
\end{aligned} \tag{3.10}$$

The R code of model (3.10) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
library(xlsxjars)
#reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# defining a matrix to be filled with input and output data
f.con=matrix(ncol=N+1,nrow=m+s)
# defining inputs and outputs to be used for deriving targets
inputs=data.frame(df[1:2])
outputs=data.frame(df[3:4])
#number of elements
m=2
s=ncol(df)-m
N = nrow(df)
for (j in 1:N) {
#defining right hand side, directions, and objective
f.rhs = c(unlist(unname(df[j,(1):(m)])),rep(0,s), 1)
f.dir = c(rep("<=",m),rep(">=",s), "=")
f.obj = c(1, rep(0,N))
# defining matrix of coefficient
for(i in 1:m) {
f.con[i,1:(N+1)]=c(0,df[,i]) }
for(r in (m+1):(s+m)) {
f.con[r,1:(N+1)]=c(as.numeric(-df[j,r]),as.numeric(df[,r])) }
```

```
#solving model
results = lp('max', as.numeric(f.obj), f.con, f.dir, f.rhs, scale=0, compute.
sens=F)
if (j==1) {
weights = results$solution[1]
lambdas = results$solution[seq(2,(N+1))]
xbench = lambdas %*% as.matrix(inputs)
ybench = lambdas %*% as.matrix(outputs)
} else {
weights = rbind(weights, results$solution[1])
lambdas = rbind(lambdas, results$solution[seq(2,(N+1))])
xbench = lambdas %*% as.matrix(inputs)
ybench = lambdas %*% as.matrix(outputs) }
}
Es = data.frame(weights, lambdas, xbench, ybench)
colnames(Es) = c('effi', rep("lambda",N), rep("x benchmark", m), rep("y
benchmark", s))
WriteXLS(Es,"5-CCR-ENV-OUT.xls", row.names =F, col.names = T)
```

Example 3.5 Consider the data set given in Table 3.1 with ten DMUs, two inputs, and two outputs. Results of running above code which are corresponding to the output-oriented CCR model are listed in Tables 3.11 and 3.12. EFFI shows the efficiency scores and Lan* shows the optimal values of intensifier variables. Tx and Ty show the coordination of target points.

3.3.2 Output-Oriented CCR Multiplier Model with R Code

Output-oriented CCR multiplier model is presented by Charnes et al. [1]. The optimal value of this model reveals the relative efficiency score of the unit under assessment (DMU_p). As this model and the previous one are a pair of primal and dual problems thus having finite optimal value for the objective function they have the same values. Vector and matrix forms of this model can be stated as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} v_i x_{ip} \\ \text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ij} - \sum_{r=1}^{s} u_r y_{rj} \geq 0, \quad j = 1, \ldots, n, \\ & \sum_{r=1}^{s} u_r y_{rp} = 1, \\ & u_r \geq 0, \quad v_i \geq 0, \qquad r = 1, \ldots, s, \quad i = 1, \ldots, m. \end{aligned} \tag{3.11}$$

Table 3.11 Efficiency scores and optimal values

DMU	EFFI	Lan_1*	Lan_2*	Lan_3*	Lan_4*	Lan_5*	Lan_6*	Lan_7*	Lan_8*	Lan_9*	Lan_{10}*
1	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	1.313	1.040	0.000	0.000	0.000	0.000	0.000	0.000	0.112	0.000	0.000
3	1.040	1.477	0.000	0.000	0.000	0.000	0.000	0.000	0.118	0.000	0.000
4	1.600	1.333	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	1.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
6	2.274	0.820	0.000	0.000	0.000	0.205	0.000	0.000	0.000	0.000	0.000
7	1.120	0.933	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
9	1.227	1.121	0.000	0.000	0.000	0.455	0.000	0.000	0.000	0.000	0.000
10	4.960	3.370	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 3.12 Target points

DMU	Tx_1	Tx_2	Ty_1	Ty_2
1	15.000	27.000	45.000	78.000
2	24.000	32.000	57.776	89.632
3	31.000	44.000	78.014	124.148
4	20.000	36.000	60.000	104.000
5	18.000	83.000	86.000	76.000
6	16.000	39.193	54.573	79.585
7	14.000	25.200	42.000	72.800
8	75.000	35.000	98.000	76.000
9	25.000	68.000	89.545	122.000
10	50.556	91.000	151.667	262.889

$$\min \quad \left[x_{1p},...,x_{mp},0,...,0\right]\left[v_1,...,v_m,u_1,...,u_s\right]^t$$

$$s.t.\left[\begin{array}{ccc:ccc} x_{11} & \cdots & x_{m1} & -y_{11} & \cdots & -y_{s1} \\ \vdots & & \vdots & \vdots & & \vdots \\ x_{1n} & \cdots & x_{mn} & -y_{1n} & \cdots & -y_{sn} \\ \hdashline 0 & \cdots & 0 & y_p & \cdots & y_{sp} \end{array}\right]\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \end{array}\right] \begin{matrix} \geq_n \\ =_1 \end{matrix} \left[\begin{array}{c} 0_{n\times 1} \\ \hdashline 1 \end{array}\right] \tag{3.12}$$

$$u_r \geq 0, \quad v_i \geq 0, \quad r=1,...,s, \quad i=1,...,m.$$

The R code of model (3.12) is as follows.

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
#number of elements
m=2
s=ncol(df)-m
N = nrow(df)
#defining right handside, and rirections
f.rhs = c(rep(0,N),1)
f.dir = c(rep(">=",N),"=")
```

```
for (j in 1:N) {
#defining objevtive
f.obj = c(df[j,(1):(m)], rep(0,s))
#defining matrix of coeficient
con1 = cbind(df[,1:m],-df[,(m+1):(m+s)])
con2 = c(rep(0,m), as.numeric(df[j,(m+1):(m+s)]))
f.con = rbind(con1,con2)
#solving model
results = lp ("min",f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=TRUE)
if (j==1) {
weights = results$solution
effcrs = results$objval
lambdas = results$duals[seq(1,N+1)]
{ else {
weights = rbind(weights, results$solution)
effcrs = rbind(effcrs, results$objval)
lambdas = rbind(lambdas, results$duals[seq(1,N+1)]) }
}
ls = data.frame(effcrs, weights,lambdas)
colnames(ls) = c('effi', "v", "u", "u", "u", rep("lambda", N), "teta");ls
WriteXLS(ls,"6-CCR-MUL-OUT.xls", row.names = F, col.names = T)
```

Example 3.6 Consider the data set given in Table 3.1. Results of running above code correspond to the output-oriented CCR model are listed in Tables 3.13 and 3.14. EFFI shows the efficiency scores and v* and u* show the optimal values of weights. Lan* which is the optimal values of intensifier variables obtained from corresponding dual model. Here θ* shows the optimal value of dual variable in enveloping form of CCR model which is equal to EFFI values.

Table 3.13 Efficiency scores and optimal solutions

DMU	Opt. obj	EFFI	v_1*	v_2*	u_1*	u_2*
1	1.000	1.000	0.000	0.037	0.000	0.013
2	1.313	0.762	0.016	0.029	0.023	0.000
3	1.040	0.962	0.010	0.017	0.013	0.000
4	1.600	0.625	0.080	0.000	0.000	0.015
5	1.000	1.000	0.022	0.007	0.012	0.000
6	2.274	0.440	0.142	0.000	0.011	0.021
7	1.120	0.893	0.080	0.000	0.000	0.015
8	1.000	1.000	0.007	0.013	0.010	0.000
9	1.227	0.815	0.026	0.009	0.014	0.000
10	4.960	0.202	0.000	0.055	0.000	0.019

Table 3.14 Optimal values of dual variables

DMU	Lan_1*	Lan_2*	Lan_3*	Lan_4*	Lan_5*	Lan_6*	Lan_7*	Lan_8*	Lan_9*	Lan_{10}*	θ*
1	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000
2	1.040	0.000	0.000	0.000	0.000	0.000	0.000	0.112	0.000	0.000	1.313
3	1.477	0.000	0.000	0.000	0.000	0.000	0.000	0.118	0.000	0.000	1.040
4	1.333	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.600
5	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	1.000
6	0.820	0.000	0.000	0.000	0.205	0.000	0.000	0.000	0.000	0.000	2.274
7	0.933	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.120
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	1.000
9	1.121	0.000	0.000	0.000	0.455	0.000	0.000	0.000	0.000	0.000	1.227
10	3.370	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	4.960

3.3.3 Output-Oriented BCC Multiplier Model with R Code

Output-oriented BCC multiplier model has been introduced by Banker et al. [2]. Relative efficiency score of the DMU_p is obtained from optimal objective function value. Note that free variable v_0 is replaced by subtraction of two non-negative variables, v_0^+ and v_0^-.

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m} v_i x_{ip} + v_0^+ - v_0^- \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ij} - \sum_{r=1}^{s} u_r y_{rj} + v_0^+ - v_0^- \geq 0, \qquad j = 1, \ldots, n, \\
& \sum_{r=1}^{s} u_r y_{rp} = 1, \\
& u_r \geq 0, \quad v_i \geq 0, \quad r = 1, \ldots, s, \quad i = 1, \ldots, m, \\
& v_0^+ \geq 0, \quad v_0^- \geq 0.
\end{aligned} \tag{3.13}$$

$$\min \left[x_{1p}, \ldots, x_{mp}, 0, \ldots, 0, 1, -1 \right] \left[v_1, \ldots, v_m, u_1, \ldots, u_s, \mathrm{v}_0^+, \mathrm{v}_0^- \right]^t$$

$$s.t. \quad \left[\begin{array}{ccc|ccc|cc} -x_{11} & \cdots & -x_{m1} & y_{11} & \cdots & y_{s1} & 1 & -1 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots \\ -x_{1n} & \cdots & -x_{mn} & y_{1n} & \cdots & y_{sn} & 1 & -1 \\ \hline 0 & \cdots & 0 & y_{1p} & \cdots & y_{sp} & 0 & 0 \end{array}\right] \begin{array}{c} \leq_n \\ =_1 \end{array} \left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hline u_1 \\ \vdots \\ u_s \\ \hline v_0^+ \\ v_0^+ \end{array}\right] \left[\begin{array}{c} 0_{n\times 1} \\ \hline 1 \end{array}\right] \tag{3.14}$$

$$\begin{aligned}
& u_r \geq 0, \quad v_i \geq 0, \quad r = 1, \ldots, s, \quad i = 1, \ldots, m, \\
& v^+{}_0 \geq 0, \quad v^-{}_0 \geq 0.
\end{aligned}$$

The R code of model (3.14) is as follows.

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
```

```
df=data.frame(read_excel(path = "data1.xlsx", sheet= "1"))
#number of elements
m=2
s=ncol(df)-m
N = nrow(df)
#defining right handside, directions
f.rhs = c(rep(0,N),1)
f.dir = c(rep(">=",N),"=")
for (j in 1:N) {
#defining objective
f.obj = c(as.numeric(df[j,(1):(m)]), rep(0,s), 1, -1)
#defining matrix of coefficient
con1 = cbind(df[,1:m],-df[,(m+1):(m+s)], matrix(1,N,1), matrix((-1),N,1))
con2= c(rep(0,m), as.numeric(df[j,(m+1):(m+s)]), 0,0)
f.con = rbind(con1, con2)
#solving model
resultss = lp ("min",f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=TRUE)
multiplierss = resultss$solution
u0 = multiplierss[s+m+1]-multiplierss[s+m+2]
f (j==1) {
weightss = c(multiplierss[seq(1,s+m)],u0)
effvrss = resultss$objval
lambdass= resultss$duals[seq(1,N+1)]
} else {
weightss = rbind(weightss,c(multiplierss[seq(1,s+m)],u0))
effvrss = rbind(effvrss, resultss$objval)
lambdass = rbind(lambdass,resultss$duals[seq(1,N+1)]) }
}
ls = data.frame(effvrss, weightss, lambdass)
colnames(ls) = c('effi', rep('v', m), rep('u', s), 'u0', rep("lambda", N),
"teta");ls
WriteXLS(ls,"7-BCC-MUL-OUT.xls", row.names = F, col.names = F)
```

Example 3.7 Consider the data set given in Table 3.1 with ten DMUs, two inputs, and two outputs. Results of running above code corresponding to the output-oriented BCC model are listed in Tables 3.15 and 3.16. EFFI shows the efficiency scores and v*, u*, and u0* show the optimal values of weights. Lan* which is the optimal values of intensifier variables obtained from corresponding dual model. Tx and Ty show the target points.

Table 3.15 Efficiency scores and optimal weights

DMU	Opt. obj	EFFI	v_1*	v_2*	u_1*	u_2*	u_0*
1	1.000	1.000	0.000	0.037	0.000	0.013	0.000
2	1.283	0.779	0.016	0.014	0.020	0.002	0.461
3	1.000	1.000	0.010	0.009	0.013	0.001	0.296
4	1.200	0.833	0.000	0.000	0.000	0.015	1.200
5	1.000	1.000	0.008	0.007	0.011	0.001	0.244
6	2.217	0.451	0.000	0.000	0.001	0.028	2.213
7	1.000	1.000	0.200	0.000	0.000	0.015	-1.800
8	1.000	1.000	0.007	0.006	0.009	0.001	0.216
9	1.137	0.879	0.009	0.007	0.014	0.000	0.468
10	1.472	0.679	0.000	0.000	0.000	0.019	1.472

Table 3.16 Optimal values of intensifier variables

DMU	Lan_1*	Lan_2*	Lan_3*	Lan_4*	Lan_5*	Lan_6*	Lan_7*	Lan_8*	Lan_9*	Lan_{10}*	Phi*
1	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.00
2	0.710	0.000	0.157	0.000	0.026	0.000	0.000	0.107	0.000	0.000	1.28
3	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.00
4	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.20
5	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	1.00
6	0.802	0.000	0.000	0.000	0.191	0.000	0.000	0.007	0.000	0.000	2.21
7	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.00
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	1.00
9	0.000	0.000	0.325	0.000	0.627	0.000	0.000	0.049	0.000	0.000	1.13
10	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.47

3.3.4 Output-Oriented BCC Envelopment Model with R Code

Output-oriented BCC envelopment model has been introduced by Banker et al. [2]. The optimal objective value shows the relative efficiency of decision making unit under evaluation (DMU_p). In accordance to the strong duality theorem, as this model and previous one are a pair of primal and dual problems thus in finite optimal solution they have the same scores. In the following the vector and matrix forms of this model are stated.

$$\begin{aligned}
\max \quad & \varphi \\
& \sum_{j=1}^{n} \lambda_i x_{ij} \le x_{ip}, && i = 1, \ldots, m, \\
& -\varphi y_{rp} + \sum_{j=1}^{u} \lambda_i y_{rj} \ge 0, && r = 1, \ldots, s, \\
& \sum_{j=1}^{n} \lambda_j = 1, \\
& \lambda_j \ge 0, && j = 1, \ldots, n.
\end{aligned} \tag{3.15}$$

$$\begin{aligned}
&\max \quad [1, 0, ..., 0][\varphi, \lambda_1, ..., \lambda_n]^t \\
&s.t. \quad \left[\begin{array}{c|ccc} 0 & x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & & \vdots \\ 0 & x_{m1} & \cdots & x_{mn} \\ \hline -y_{1p} & y_{11} & \cdots & y_{1n} \\ \vdots & \vdots & & \vdots \\ -y_p & y_{s1} & \cdots & y_{sn} \\ \hline 0 & 1 & \cdots & 1 \end{array}\right] \left[\begin{array}{c} \varphi \\ \hline \lambda_1 \\ \vdots \\ \lambda_n \end{array}\right] \begin{array}{c} \le_m \\ \hline \ge_s \\ =_1 \end{array} \left[\begin{array}{c} x_{1p} \\ \vdots \\ x_{mp} \\ \hline o_{m\times 1} \\ \hline 1 \end{array}\right] \\
&\lambda_j \ge 0, \qquad\qquad j = 1, ..., n.
\end{aligned} \tag{3.16}$$

The R code of model (3.16) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
library(xlsxjars)
# reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# defining a matrix to be filled with input and output data
f.con=matrix(ncol=N+1,nrow=m+s+1)
# defining inputs and outputs to be used for deriving targets
inputs=data.frame(df[1:2])
outputs=data.frame(df[3:4])
```

```
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
for (j in 1:N) {
# defining right hand side, directions, objective
f.rhs = c(unlist(unname(df[j,(1):(m)])),rep(0,s), 1)
f.dir = c(rep("<=",m),rep(">=",s), "=")
f.obj = c(1, rep(0,N))
# defining matrix of coeficient
for(i in 1:m) {
f.con[i,1:(N+1)]=c(0,df[,i]) }
for(r in (m+1):(s+m)) {
f.con[r,1:(N+1)]=c(as.numeric(-df[j,r]),as.numeric(df[,r])) }
f.con[m+s+1, 1:(N+1)]=c(0, rep(1,N))
# solving model
results = lp('max', as.numeric(f.obj), f.con, f.dir, f.rhs, scale=0, compute.
sens=TRUE)
if (j==1) {
weights = results$solution[1]
lambdas = results$solution[seq(2,(N+1))]
xbench = lambdas %*% as.matrix(inputs)
ybench = lambdas %*% as.matrix(outputs)
} else {
weights = rbind(weights, results$solution[1])
lambdas = rbind(lambdas, results$solution[seq(2,(N+1))])
xbench = lambdas %*% as.matrix(inputs)
ybench = lambdas %*% as.matrix(outputs) } }
Es = data.frame(weights, lambdas, xbench, ybench)
colnames(Es) = c('effi', rep("lambda",N), rep("x benchmark", m), rep("y
benchmark", s))
WriteXLS(Es,"8-BCC-ENV-OUT.xls", row.names = F, col.names = T)
```

Example 3.8 Consider the data set given in Table 3.1 with ten DMUs, two inputs, and two outputs. Results of running above code corresponding to the output-oriented BCC model are listed in Tables 3.17 and 3.18. EFFI shows the efficiency scores and Lan* shows the optimal values of intensifier variables. Tx and Ty show the target points.

Table 3.17 optimal solutions

DMU	Lan_1*	Lan_2*	Lan_3*	Lan_4*	Lan_5*	Lan_6*	Lan_7*	Lan_8*	Lan_9*	Lan_{10}*
1	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.710	0.000	0.157	0.000	0.026	0.000	0.000	0.107	0.000	0.000
3	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
6	0.802	0.000	0.000	0.000	0.191	0.000	0.000	0.007	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
9	0.000	0.000	0.325	0.000	0.627	0.000	0.000	0.049	0.000	0.000
10	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 3.18 Target points and efficiency

DMU	Tx_1	Tx_2	Ty_1	Ty_2	Opt. obj	EFFI
1	15.000	27.000	45.000	78.000	1.000	1.000
2	24.000	32.000	56.450	69.279	1.283	0.779
3	31.000	44.000	75.000	24.000	1.000	1.000
4	15.000	27.000	45.000	78.000	1.200	0.833
5	18.000	83.000	86.000	76.000	1.000	1.000
6	16.000	37.761	53.214	77.603	2.217	0.451
7	14.000	56.000	24.000	65.000	1.000	1.000
8	75.000	35.000	98.000	76.000	1.000	1.000
9	25.000	68.000	83.015	59.122	1.137	0.880
10	15.000	27.000	45.000	78.000	1.472	0.679

3.4 Additive DEA Models with R Codes

In this section, R codes for additive DEA model are presented.

3.4.1 Additive CCR Model with R Code

Additive model has been introduced by Bardhan et al. [3]. The aim of this model is to maximize sum of slack variables. According to the optimal objective function value, the set of DMUs can be classified into two subsets of efficient and inefficient according to the zero or positive values of objective values. The vector and matrix forms of this model can be stated as follows:

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{m} s_i^- + \sum_{r=1}^{s} s_r^+ \\
\text{s.t.} \quad & \sum_{j=1}^{n} \lambda_i x_{ij} + s_i^- = x_{ip}, \qquad i = 1\ldots, m, \\
& \sum_{j=1}^{n} \lambda_i y_{rj} - s_r^+ = y_{rp}, \quad r = 1, \ldots, s, \\
& \lambda_j \geq 0, \qquad\qquad\qquad j = 1, \ldots, n, \\
& s_i^- \geq 0, s_r^+ \geq 0, \qquad i = 1\ldots, m, r = 1, \ldots, s.
\end{aligned}
\tag{3.17}
$$

$$
\max \quad \left[0,\ldots,0,\underbrace{1,\ldots,1}_{m},\underbrace{1,\ldots,1}_{s}\right]\left[\lambda_1,\ldots,\lambda_n,s_1^-,\ldots,s_m^-,s_1^+,\ldots,s_s^+\right]^t
$$

$$
s.t. \begin{bmatrix} x_{11} & \cdots & x_{1n} & & \\ \vdots & & \vdots & I_m & o_{m\times s} \\ x_{m1} & \cdots & x_{mn} & & \\ \hline y_{11} & \cdots & y_{1n} & & \\ \vdots & & \vdots & o_{s\times m} & -I_s \\ y_{s1} & \cdots & y_{sn} & & \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \hline s_1^- \\ \vdots \\ s_m^- \\ \hline s_1^+ \\ \vdots \\ s_s^+ \end{bmatrix} \begin{matrix} \\ \\ =_m \\ \hline \\ \\ =_s \\ \\ \\ \end{matrix} \begin{bmatrix} x_{1p} \\ \vdots \\ x_{mp} \\ y_{1p} \\ \vdots \\ y_{sp} \end{bmatrix}
\tag{3.18}
$$

$$
s_i^- \geq 0, s_r^+ \geq 0, \qquad i = 1\ldots, m, r = 1, \ldots, s.
$$

The R code of model (3.18) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
```

```
# defining a matrix to be filled with input and output data
f.con=matrix(ncol=N,nrow=m+s)
for (j in 1:N) {
# defining right hand side, directions, objective
f.rhs = c(as.numeric(df[j,1:m]), as.numeric(df[j,(m+1):(m+s)]))
f.dir = c(rep("=",m), rep("=", s))
f.obj = c(rep(0,N), rep(1,m), rep(1, s))
# defining matrix of coefficient
for(i in 1:m) { f.con[i,1:N]=c(df[,i]) }
for(r in (m+1):(s+m)) { f.con[r,1:N]=c(df[,r]) }
c1=rbind(diag(m), matrix(0,s,m))
c2=rbind(matrix(0,m,s),-diag(s))
final.con = cbind(f.con, c1, c2)
# solving model
results = lp('max', as.numeric(f.obj), final.con, f.dir, f.rhs, scale=0, compute.
sens=F)
if (j==1) {
weights = results$solution[seq(1:(N+m+s))]
effcrs = results$objval
} else {
weights = rbind(weights, results$solution[seq(1:(N+m+s))])
effcrs = rbind(effcrs, results$objval) } }
EEs = data.frame(effcrs, weights)
colnames(EEs) = c('sum', rep("lambda", N), rep('s-', m), rep('s+', s))
WriteXLS(Es,"9-additive-CCR.xls", row.names = F, col.names = T)
```

Example 3.9 Consider the data set given in Table 3.1. In constant returns to scale form of the technology the results of additive mode are listed in Tables 3.19 and 3.20. Sum shows the optimal objective function and (Lan*) show the optimal values of intensifier variables. Also, optimal values of slack variables are reported in Table 3.20.

3.4.2 Additive BCC Model with R Code

In a similar way, the vector and matrix forms of additive model with variable returns to scale are stated as follows:

Table 3.19 Optimal values

DMU	Sum	Lan_1*	Lan_2*	Lan_3*	Lan_4*	Lan_5*	Lan_6*	Lan_7*	Lan_8*	Lan_9*	Lan_{10}*
1	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	54.000	1.185	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	106.277	1.575	0.000	0.000	0.000	0.000	0.000	0.000	0.042	0.000	0.000
4	83.000	1.333	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
6	125.400	1.067	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
7	56.600	0.933	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
9	109.000	1.667	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
10	359.000	3.370	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 3.20 Optimal values of slacks

DMU	S_1^-*	S_2^-*	S_1^+*	S_2^+*
1	0.000	0.000	0.000	0.000
2	6.222	0.000	9.333	38.444
3	4.221	0.000	0.000	102.056
4	0.000	21.000	23.000	39.000
5	0.000	0.000	0.000	0.000
6	0.000	53.200	24.000	48.200
7	0.000	30.800	18.000	7.800
8	0.000	0.000	0.000	0.000
9	0.000	23.000	2.000	84.000
10	21.444	0.000	127.667	209.889

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{m} s_i^- + \sum_{r=1}^{s} s_r^+ \\
s.t. \quad & \sum_{j=1}^{n} \lambda_i x_{ij} + s_i^- = x_{ip}, \qquad i = 1, \ldots, m, \\
& \sum_{j=1}^{n} \lambda_i y_{rj} - s_i^+ = y_{rp}, \quad r = 1, \ldots, s, \\
& \sum_{j=1}^{n} \lambda_i = 1, \\
& \lambda_j \geq 0, \qquad\qquad j = 1, \ldots, n, \\
& s_i^- \geq 0, s_r^+ \geq 0, \qquad i = 1 \ldots, m, r = 1, \ldots, s.
\end{aligned}
\tag{3.19}
$$

$$
\begin{aligned}
\max \quad & \left[0, \ldots, 0, \underbrace{1, \ldots, 1}_{m}, \underbrace{1, \ldots, 1}_{s}\right] \left[\lambda_1, \ldots, \lambda_n, s_1^-, \ldots, s_m^-, s_1^+, \ldots, s_s^+\right]^t \\
s.t. \quad & \left[\begin{array}{ccc|c|c} x_{11} & \cdots & x_{1n} & & \\ \vdots & & \vdots & I_m & o_{m\times s} \\ x_{m1} & \cdots & x_{mn} & & \\ \hline y_{11} & \cdots & y_{1n} & & \\ \vdots & & \vdots & o_{s\times m} & -I_s \\ y_{s1} & \cdots & y_{sn} & & \\ \hline 1 & \ldots & 1 & o_{1\times m} & o_{1\times s} \end{array}\right] \left[\begin{array}{c} \lambda_1 \\ \vdots \\ \lambda_n \\ \hline s_1^- \\ \vdots \\ s_m^- \\ \hline s_1^+ \\ \vdots \\ s_s^+ \end{array}\right] =_{m+s+1} \left[\begin{array}{c} x_{1p} \\ \vdots \\ x_{mp} \\ \hline y_{1p} \\ \vdots \\ y_{sp} \\ \hline 1 \end{array}\right] \\
& \lambda_j \geq 0, \qquad j = 1, \ldots, n, \\
& s_i^- \geq 0, s_r^+ \geq 0, \qquad i = 1 \ldots, m, r = 1, \ldots, s.
\end{aligned}
\tag{3.20}
$$

The R code of model (3.20) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
# defining a matrix to be filled with input and output data
f.con=matrix(ncol=N,nrow=m+s+1)
# considering inputs and outputs to be used for deriving target
inputs = df(df[,1:m])
outputs = df[,(m+1):(s+m)]
for (j in 1:N) {
# defining right hand side, directions, objective
f.rhs = c(as.numeric(df[j,1:m]), as.numeric(df[j,(m+1):(m+s)]), 1)
f.dir = c(rep("=",m), rep("=", s), "=")
f.obj = c(rep(0,N), rep(1,m), rep(1, s))
# defining matrix of coefficient
for(i in 1:m) { f.con[i,1:N]=c(as.numeric(df[,i])) }
for(r in (m+1):(s+m))
{ f.con[r,1:N]=c(as.numeric(df[,r])) }
c1=rbind(diag(m), matrix(0,s,m),0)
c2=rbind(matrix(0,m,s),-diag(s),0)
f.con[m+s+1, 1:N]=c(rep(1,N))
final.con = cbind(f.con, c1, c2)
# solving model
results = lp('max', as.numeric(f.obj), final.con, f.dir, f.rhs, scale=0, compute.
sens=TRUE)
if (j==1) {
weights = results$solution[seq(1:(N+m+s))]
effcrs = results$objval
} else {
weights = rbind(weights, results$solution[seq(1:(N+m+s))])
effcrs = rbind(effcrs, results$objval) } }
EEs = data.frame(effcrs, weights)
colnames(EEs) = c('sum', rep("lambda", N), rep('s-', m), rep('s+', s))
WriteXLS(Es,"10-additive-BCC.xls", row.names = F, col.names = T)
```

Table 3.21 Optimal objective function of additive model

DMU	Sum	Lan_1*	Lan_2*	Lan_3*	Lan_4*	Lan_5*	Lan_6*	Lan_7*	Lan_8*	Lan_9*	Lan_{10}*
1	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	39.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	56.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
6	120.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
9	42.596	0.359	0.000	0.000	0.000	0.500	0.000	0.000	0.142	0.000	0.000
10	167.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 3.22 Optimal values of slacks

DMU	s_1^-*	s_2^-*	s_1^+*	s_2^+*
1	0.000	0.000	0.000	0.000
2	9.000	5.000	1.000	24.000
3	0.000	0.000	0.000	0.000
4	5.000	30.000	8.000	13.000
5	0.000	0.000	0.000	0.000
6	1.000	55.000	21.000	43.000
7	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000
9	0.000	11.879	0.000	30.717
10	57.000	64.000	21.000	25.000

Example 3.10 Consider the data given Table 3.1 with ten DMUs each uses two inputs to produce two outputs. The results of the following code corresponding to the additive model with variable returns to scale technology are reported in Table 3.21. Sum shows the optimal objective function of the additive model (3.20) and (lan*) shows the optimal values of intensifier variables. Table 3.22 shows the optimal values of slack variables.

3.5 R Codes for Input-Oriented DEA Multiplier Model with ε

In this section, R codes for input-oriented DEA multiplier with ε are presented.

3.5.1 R Code for Input-Oriented BCC Multiplier Model with ε

Input-oriented BCC multiplier model with ε has been introduced by Charnes et al. [4]. The optimal objective value shows the relative efficiency score of DMUs. In this model the free variable u_0 is considered as the subtraction of two non-negative variables, u_0^- and u_0^+. The vector and matrix forms of this model is as follows.

$$
\begin{aligned}
\max \quad & \sum_{r=1}^{s} u_r y_{rp} + u_0^+ - u_0^- \\
\text{s.t.} \quad & -\sum_{i=1}^{m} v_i x_{ij} + \sum_{r=1}^{s} u_r y_{ij} + u_0^+ - u_0^- \le 0, \qquad j = 1, \ldots, n, \\
& \sum_{i=1}^{m} v_r x_{ip} = 1 \\
& v_i \ge \varepsilon, u_r \ge \varepsilon, \quad i = 1, \ldots, m, r = 1, \ldots, s, \\
& u_0^+ \ge 0, u_0^- \ge 0.
\end{aligned}
\tag{3.21}
$$

$$
\begin{aligned}
&\max \quad \left[0,\ldots,0,y_{1p},\ldots,y_{sp},1,-1\right]\left[v_1,\ldots,v_m,u_1,\ldots,u_s,u_0^+,u_0^-\right]^t \\
&s.t. \\
&\left[\begin{array}{ccc|ccc|cc}
x_{11} & \cdots & x_{1n} & y_{11} & \cdots & y_{1s} & 1 & -1 \\
\vdots & & \vdots & \vdots & & \vdots & & \\
x_{m1} & \cdots & x_{mn} & y_{1n} & \cdots & y_{sn} & 1 & -1 \\
\hline
x_{1p} & \cdots & x_{mp} & 0 & \ldots & 0 & 0 & 0 \\
\hline
\multicolumn{3}{c|}{I_m} & \multicolumn{3}{c|}{0} & \multicolumn{2}{c}{o_{m\times 2}} \\
\hline
\multicolumn{3}{c|}{o_{s\times n}} & \multicolumn{3}{c|}{I_s} & \multicolumn{2}{c}{o_{m\times 2}} \\
\hline
\multicolumn{3}{c|}{o_{n\times 2}} & \multicolumn{3}{c|}{o_{2\times s}} & \multicolumn{2}{c}{I_2}
\end{array}\right]
\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hline u_1 \\ \vdots \\ u_s \\ \hline u_0^+ \\ \hline u_0^- \end{array}\right]
\begin{array}{c} \le_n \\ \hline =_1 \\ \hline \ge_{m+s} \\ \hline \ge_2 \end{array}
\left[\begin{array}{c} \\ 0_{n\times 1} \\ \hline 1 \\ \hline \varepsilon_{m+s} \\ \hline 0_{2\times 1} \end{array}\right] \\
&u_0^+ \ge 0,\ u_0^- \ge 0.
\end{aligned}
\tag{3.22}
$$

The R code of model (3.22) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df = data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# number of elements
m=2
s = ncol(df)-m
N = nrow(df)
inputs = data.matrix(df[1:2])
outputs = data.matrix(df[3:4])
# defining right hand side, directions,
f.rhs = c(rep(0,N),1, rep(0.00001,m), rep(0.00001,s), 0., 0.)
f.dir = c(rep("<=",N),"=", rep(">=",(m+s+2)))
for (j in 1:N) { f.obj = c(rep(0,m),outputs[j,], 1, -1)
```

```
con1 = cbind(-inputs, outputs, 1, -1)
con2=c(inputs[j,], rep(0,s), 0, 0)
con3=cbind(diag(m), matrix(0,m,s), 0, 0)
con4=cbind(matrix(0,s,m), diag(s), 0,0)
con5=cbind(matrix(0,2,(m+s)), diag(2))
f.con = rbind(con1, con2, con3, con4, con5)
# solving model
result = lp ("max",f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=TRUE)
multiplier = result$solution
u0 = multiplier[s+m+1]-multiplier[s+m+2]
if (j==1) {
weight = c(multiplier[seq(1,s+m)],u0)
effEPS = result$objval
} else {
weight = rbind(weight,c(multiplier[seq(1,s+m)],u0))
effEPS = rbind(effEPS, result$objval) } }
EP = data.frame(effEPS, weight)
colnames(EP) = c('effi', rep('v', m), rep('u', s), 'u0')
WriteXLS(EP,"11-BCC-MUL-EPSILON.xls", row.names = F, col.names = T)
```

Example 3.11 Consider the data set consists of ten DMUs with two inputs and two outputs given in Table 3.1. The efficiency score obtained from input-oriented BCC model (EFFI) as well as optimal weights (v*, u* and u0*) are listed in Table 3.23.

Table 3.23 Efficiency scores and optimal weights

DMU	EFFI.	v_1*	v_2*	u_1*	u_2*	u_0*
1	1.000	0.000	0.037	0.000	0.013	0.000
2	0.843	0.000	0.031	0.000	0.000	0.842
3	1.000	0.011	0.015	0.014	0.000	−0.074
4	0.731	0.050	0.000	0.002	0.000	0.643
5	1.000	0.022	0.007	0.012	0.000	0.000
6	0.875	0.062	0.000	0.003	0.000	0.803
7	1.000	0.071	0.000	0.000	0.005	0.645
8	1.000	0.007	0.013	0.010	0.000	0.000
9	0.851	0.016	0.009	0.013	0.000	−0.128
10	0.296	0.000	0.011	0.000	0.000	0.295

3.5.2 R Code for Input-Oriented CCR Multiplier Model with ε

Input-oriented CCR multiplier model with ε has been introduced by Charnes et al. [4]. According to this model the relative efficiency score of DMUs can be obtained. The vector and matrix forms of this model are as follows:

$$\begin{aligned}
\max \quad & \sum_{r=1}^{s} u_r y_{rp} \\
\text{s.t.} \quad & -\sum_{i=1}^{m} v_i x_{ij} + \sum_{r=1}^{s} u_r y_{rj} \le 0, \quad j = 1, \ldots, n, \\
& \sum_{i=1}^{m} v_j x_{ip} = 1, \\
& v_i \ge \varepsilon, \qquad\qquad i = 1, \ldots, m, \\
& u_r \ge \varepsilon, \qquad\qquad r = 1, \ldots, s.
\end{aligned} \tag{3.23}$$

$$\begin{aligned}
\max \quad & \left[0, \ldots, 0, y_{1p}, \ldots, y_{sp}\right]\left[v_1, \ldots, v_m, u_1, \ldots, u_s\right]^t \\
s.t. \quad & \left[\begin{array}{ccc|ccc}
-x_{11} & \cdots & -x_{m1} & y_{11} & \cdots & -y_{s1} \\
\vdots & & \vdots & \vdots & & \vdots \\
-x_{1n} & \cdots & -x_{mn} & y_{1n} & \cdots & y_{sn} \\
x_{1p} & \cdots & x_{mp} & 0 & \cdots & 0 \\
\hline
\multicolumn{3}{c|}{I_m} & \multicolumn{3}{c}{o_{m\times s}} \\
\hline
\multicolumn{3}{c|}{o_{s\times m}} & \multicolumn{3}{c}{I_s}
\end{array}\right]\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hline u_1 \\ \vdots \\ u_s \end{array}\right] \begin{array}{c} \le_n \\ =_1 \\ \ge_{m+s} \end{array} \left[\begin{array}{c} 0_{n\times 1} \\ \hline 1 \\ \hline \varepsilon_{m+s} \end{array}\right] \\
& v_i \ge \varepsilon, \qquad i = 1, \ldots, m, \\
& u_r \ge \varepsilon, \qquad r = 1, \ldots, s.
\end{aligned} \tag{3.24}$$

The R code of model (3.24) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df = data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# number of elements
m=2
s = ncol(df)-m
N = nrow(df)
```

```
inputs = data.matrix(df[1:2])
outputs = data.matrix(df[3:4])
# defining right hand side, directions
f.rhs = c(rep(0,N),1, rep(0.00001,m), rep(0.00001,s))
f.dir = c(rep("<=",N),"=", rep(">=",(m+s)))
for (j in 1:N) {
# defining objective, and matrix of coefficient
f.obj = c(rep(0,m),outputs[j,])
con1 = cbind(-inputs, outputs)
con2=c(inputs[j,], rep(0,s))
con3=cbind(diag(m), matrix(0,m,s))
con4=cbind(matrix(0,s,m), diag(s))
f.con = rbind(con1, con2, con3, con4)
# solving model
resultss = lp ("max",f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weightss = c(results$solution[seq(1,s+m)])
effvrss = resultss$objval
} else {
weightss = rbind(weightss,c(results$solution [seq(1,s+m)]))
effvrss = rbind(effvrss, resultss$objval) } }
ls = data.frame(effvrss, weightss)
colnames(ls) = c('effi', rep('v', m), rep('u', s))
WriteXLS(ls,"12-CCR-MUL-INP-2-phase.xls", row.names = F, col.names =
T)
```

Example 3.12 Consider the data set consists of ten DMUs with two inputs and two outputs given in Table 3.1. The efficiency score obtained from input-oriented CCR model (EFFI) as well as optimal weights (v* and u*) are listed in Table 3.24.

Table 3.24 Efficiency scores and optimal weights

DMU	EFFI	v_1*	v_2*	u_1*	u_2*
1	1.000	0.000	0.037	0.000	0.013
2	0.761	0.012	0.022	0.017	0.000
3	0.960	0.009	0.016	0.013	0.000
4	0.625	0.050	0.000	0.000	0.010
5	1.000	0.022	0.007	0.012	0.000
6	0.440	0.062	0.000	0.005	0.009
7	0.892	0.071	0.000	0.000	0.014
8	1.000	0.007	0.013	0.010	0.000
9	0.815	0.021	0.007	0.011	0.000
10	0.201	0.000	0.011	0.000	0.004

3.6 Two-Phase Input-Oriented DEA Envelopment Model with R Code

In this section, R codes for two-phase input-oriented DEA multiplier models are presented.

3.6.1 Two-Phase Input-Oriented BCC Envelopment Model with R Code

The two-phase input-oriented BCC envelopment model has been introduced by Charnes et al. [5]. The optimal objective function value of the first phase shows the relative efficiency of DMU_p under evaluation. Replacing θ^* in the second model, the second phase shows the optimal slack variable. According to these optimal slack variables it is possible to obtain the Pareto efficient target point for inefficient units. The vector and matrix forms of this model are as following.

$$\begin{aligned}
&\min \quad \theta \\
&\text{s.t.} \quad -\theta x_{ip} + \sum_{j=1}^{n} \lambda_i x_{ij} \le 0, \quad i = 1, \ldots, m, \\
&\qquad \sum_{j=1}^{n} \lambda_j y_{rj} \ge y_{rp}, \qquad r = 1, \ldots, s, \\
&\qquad \sum_{j=1}^{n} \lambda_j = 1, \\
&\qquad \lambda_j \ge 0, \qquad j = 1, \ldots, n,
\end{aligned} \tag{3.25}$$

$$\begin{aligned}
\max \quad & \sum_{i=1}^{m} s_i^- + \sum_{r=1}^{s} s_r^+ \\
\text{s.t.} \quad & \sum_{j=1}^{n} \lambda_j x_{rj} - s_r^- = \theta x_{ip}^*, \quad i = 1, \ldots, m \\
& \sum_{j=1}^{n} \lambda_j y_{rj} + s_r^+ = y_{rp}, \quad r = 1, \ldots, s, \\
& \sum_{j=1}^{n} \lambda_i = 1, \\
& \lambda_j \ge 0, \qquad j = 1, \ldots, n, \\
& s_i^- \ge 0, \qquad s_r^+ \ge 0, \quad i = 1, \ldots, m, r = 1, \ldots, s.
\end{aligned} \tag{3.26}$$

$$\begin{aligned}
&\min \quad [1,0,\ldots,0,][\theta,\lambda_1,\ldots,\lambda_n]^t \\
&s.t. \quad \left[\begin{array}{c|ccc} -x_{1p} & x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & & \vdots \\ -x_{mp} & x_{m1} & \cdots & x_{mn} \\ \hline 0 & y_{11} & \cdots & y_{1n} \\ \vdots & \vdots & & \vdots \\ 0 & y_{s1} & \cdots & y_{sn} \\ \hline 0 & 1 & \cdots & 1 \end{array}\right] \left[\begin{array}{c} \theta \\ \hline \lambda_1 \\ \vdots \\ \lambda_n \end{array}\right] \begin{array}{c} \leq_m \\ \hline \geq_s \\ =_1 \end{array} \left[\begin{array}{c} 0_{m\times 1} \\ \hline y_{1p} \\ \vdots \\ y_{sp} \\ \hline 1 \end{array}\right] \\
&\quad \lambda_j \geq 0, \qquad\qquad j=1,\ldots,n.
\end{aligned} \tag{3.27}$$

$$\begin{aligned}
&\max \quad [0,\ldots,0,1,\ldots,1,1,\ldots,1]\left[\lambda_1,\ldots,\lambda_n,s_1^-,\ldots,s_m^-,s_1^+,\ldots,s_r^+\right]^t \\
&s.t. \quad \left[\begin{array}{ccc|c|c} x_{11} & \cdots & x_{1n} & & \\ \vdots & & \vdots & -I_m & o_{m\times s} \\ x_{mn} & \cdots & x_{mn} & & \\ \hline y_{11} & \cdots & y_{1n} & & \\ \vdots & & \vdots & o_{s\times m} & I_s \\ y_{s1} & \cdots & y_{sn} & & \\ \hline 1 & \ldots & 1 & o_{1\times m} & o_{1\times s} \end{array}\right] \left[\begin{array}{c} \lambda_1 \\ \vdots \\ \lambda_n \\ \hline s_1^- \\ \vdots \\ s_n^- \\ \hline s_1^+ \\ \vdots \\ s_s^+ \end{array}\right] =_{m+s+1} \left[\begin{array}{c} \theta x_{ip}^* \\ \vdots \\ \theta x_{mp}^* \\ \hline y_{1p} \\ \vdots \\ y_{sp} \\ \hline 1 \end{array}\right] \\
&\quad \lambda_j \geq 0, \qquad\qquad j=1,\ldots,n, \\
&\quad s_i^- \geq 0, \quad s_r^+ \geq 0, \qquad i=1,\ldots,m, r=1,\ldots,s.
\end{aligned} \tag{3.28}$$

The R code of models (3.27) and (3.28) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
# defining a matrix to be filled with input and output data (for phase 1)
f.con=matrix(ncol=N+1,nrow=m+s+1)
for (j in 1:N) {
# defining right hand side, directions, and objective (phase 1)
```

```
f.rhs = c(rep(0,m),unlist(unname(df[j,(m+1):(m+s)])), 1)
f.dir = c(rep("<=",m),rep(">=",s), "=")
f.obj = c(1, rep(0,N))
for(i in 1:m){
# defining,atrix of coefficient (phase 1)
f.con[i,1:(N+1)]=c(as.numeric(-df[j,i]),df[,i]) }
for(r in (m+1):(s+m)){
f.con[r,1:(N+1)]=c(0,as.numeric(df[,r])) }
f.con[m+s+1, 1:(N+1)]=c(0, rep(1,N))
# solving model (phase 1)
results = lp('min', as.numeric(f.obj), f.con, f.dir, f.rhs, scale=0,compute.
sens=TRUE)
if (j==1) {
weights = results$solution[1]
eff = results$objval
lambdas = results$duals[seq(2,N+1)]
} else {
weights = rbind(weights, results$solution[1])
eff = rbind(eff, results$objval)
lambdas = rbind(lambdas, results$solution[seq(2,N+1)]) } }
Es = data.frame(eff)
# considering optimal value of θ as parameter (for phase 2)
teta_0 = unname(Es[,1])
tetx=matrix(ncol=m,nrow=N)
for(i in 1:m) {
tetx[,i]= teta_0*df[,i] }
# defining a matrix to be filled with input and output data (for phase 2)
f.con1=matrix(ncol=N+m+s,nrow=m+s+1)
# defining objective, directions (for phase 2)
f.obj1 = c(rep(0,N),rep(1,m),rep(1,s))
f.dir1 = c(rep("=",m+s+1))
for (j in 1:N) {
# defining matrix of coefficient and right hand side (for phase 2)
for(i in 1:m)
{ f.con1[i,1:N] = c(as.numeric(df[,i])) }
for(r in (m+1):(s+m))
{ f.con1[r,1:N] = c(as.numeric(df[,r])) }
f.con1[m+s+1, 1:N]=c(rep(1,N))
f.con1[, (N+1):(N+m)] = rbind(-diag(m), matrix(0,s,m),0)
f.con1[, (N+m+1):(N+m+s)]=rbind(matrix(0,m,s),diag(s),0)
f.rhs1 = c(unlist(unname(tetx[j,])),unlist(unname(data.frame(df[j,(m+1):(m
+s)]))), 1)
# solving model (for phase 2)
```

```
resuls1 = lp('max', as.numeric(f.obj1), f.con1, f.dir1, f.rhs1, scale=0,com-
pute.sens=TRUE)
if (j==1) {
weights1 = results1$solution
eff1 = results1$objval
lambdas = results$duals[seq(1,N)]
} else {
weights1 = rbind(weights1, results1$solution)
eff1 = rbind(eff1, results1$objval)
lambdas = rbind(lambdas, results$duals[seq(1,N)]) } }
Es1 = data.frame(eff1, eff)
colnames(Es1) = c('phase2', 'phase1')
WriteXLS(Es1,"13-TWO-Phase-BCC.xls", row.names = F, col.names = T)
```

Example 3.13 Considering the data set given in Table 3.1 with ten DMUs each uses two inputs to produce two outputs. According to the given code for Two-phase BCC model, the results of phase 1 and phase 2 are gathered in Table 3.25.

As "compute.sens = TRUE", thus it is possible to obtain dual variables (input and output weights of corresponding multiplier model) as well.

3.6.2 Two-Phase Input-Oriented CCR Envelopment Model with R Code

The two-phase input-oriented CCR envelopment model is introduced by Charnes et al. [5]. The first and second phase objective functions respectively shows the relative efficiency score and optimal slack variables. According to optimal solution

Table 3.25 Optimal objective function

DMU	Phase 2	Phase 1
1	167.000	1.000
2	136.750	0.844
3	0.000	1.000
4	131.717	0.731
5	147.000	1.000
6	12.250	0.875
7	105.000	1.000
8	0.000	1.000
9	107.549	0.851
10	114.637	0.297

of these two models one can obtain the target units of inefficient DMUs. Vector and matrix forms of this model are stated as follows:

$$\begin{aligned} \min \quad & \theta \\ \text{s.t.} \quad & \sum_{j=1}^{n} \lambda_i x_{ij} \le \theta x_{ip}, \quad i = 1, \ldots, m, \\ & \sum_{j=1}^{n} \lambda_j x_{rj} \ge y_{rp}, \quad r = 1, \ldots, s, \\ & \lambda_j \ge 0n \qquad j = 1, \ldots, n. \end{aligned} \tag{3.29}$$

$$\begin{aligned} \max \quad & \sum_{i=1}^{m} s_i^- + \sum_{r=1}^{s} s_r^+ \\ \text{s.t.} \quad & \sum_{j=1}^{n} \lambda_i x_{ij} - s_i^- = \theta^* x_{ip}, \quad i = 1, \ldots, m, \\ & \sum_{j=1}^{u} x_j y_{rj} + s_r^+ = y_{rp}, \quad r = 1, \ldots, s \\ & \lambda_j \ge 0, \quad j = 1, \ldots, n, \\ & s_i^- \ge 0, s_r^+ \ge 0, i = 1, \ldots, m, r = 1, \ldots, s. \end{aligned} \tag{3.30}$$

$$\begin{aligned} & \min \quad [1, 0, \ldots, 0][\theta, \lambda_1, \ldots, \lambda_n]^t \\ & s.t. \begin{bmatrix} -x_{1p} & x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & & \vdots \\ -x_{mp} & x_{mn} & \cdots & x_{mn} \\ \hline 0 & y_{11} & \cdots & \\ \vdots & \vdots & & \vdots \\ 0 & y_{s1} & \cdots & y_{sn} \end{bmatrix} \begin{bmatrix} \theta \\ \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix} \begin{matrix} \le_m \\ \ge_s \end{matrix} \begin{bmatrix} 0_{m\times 1} \\ y_{1p} \\ \vdots \\ y_{sp} \end{bmatrix} \\ & \lambda_j \ge 0, \quad j = 1, \ldots, n. \end{aligned} \tag{3.31}$$

$$\begin{aligned} & \max \quad \left[0, \ldots, 0, \underbrace{1, \ldots, 1}_{m}, \underbrace{1, \ldots, 1}_{s}\right] \left[\lambda_1, \ldots, \lambda_n, s_1^-, \ldots, s_m^-, s_1^+, \ldots, s_s^+\right]^t \\ & s.t. \begin{bmatrix} x_{11} & \cdots & x_{1n} & & \\ \vdots & & \vdots & -I_m & o_{m\times s} \\ x_{mn} & \cdots & x_{mn} & & \\ \hline y_{11} & \cdots & y_{1n} & & \\ \vdots & & \vdots & o_{s\times m} & I_s \\ y_{s1} & \cdots & y_{sn} & & \\ \hline 1 & \cdots & 1 & o_{1\times m} & o_{1\times s} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \hline s_1 \\ \vdots \\ s_n \\ \hline s_1^+ \\ \vdots \\ s_s^+ \end{bmatrix} =_{m+s+1} \begin{bmatrix} \theta x_{ip}^* \\ \vdots \\ \theta x_{mp}^* \\ \hline y_{1p} \\ \vdots \\ y_{sp} \\ \hline 1 \end{bmatrix} \\ & \lambda_j \ge 0, j = 1, \ldots, n, \\ & s_i^- \ge 0, s_r^+ \ge 0, \text{i} = 1, \ldots, \text{m}, r = 1, \ldots, s. \end{aligned} \tag{3.32}$$

The R code of models (3.31) and (3.32) is as follows.

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
#number of elements
m=2
s=ncol(df)-m
N = nrow(df)
f.con=matrix(ncol=N+1,nrow=m+s)
for (j in 1:N) {
#defining right hand side, directions, and objective(phase 1)
f.rhs = c(rep(0,m),unlist(unname(df[j,(m+1):(m+s)])))
f.dir = c(rep("<=",m),rep(">=",s))
f.obj = c(1, rep(0,N))
#defining matrix of coefficient (phase 1)
for(i in 1:m){
f.con[i,1:(N+1)]=c(as.numeric(-df[j,i]),df[,i]) }
for(r in (m+1):(s+m))
{ f.con[r,1:(N+1)]=c(0,as.numeric(df[,r])) }
#solving model (phase 1)
results = lp('min', as.numeric(f.obj), f.con, f.dir, f.rhs, scale=0, compute.
sens=F)
if (j==1) {
weights = results$solution[1]
eff = results$objval
lambdas = results$duals[seq(2,N+1)]
} else {
weights = rbind(weights, results$solution[1])
eff = rbind(eff, results$objval)
lambdas = rbind(lambdas, results$solution[seq(2,N+1)]) } }
Es = data.frame(eff)
# considering optimal value of θ as parameter (for phase 2)
teta_0 = unname(Es[,1]);teta_0
tetx=matrix(ncol=m,nrow=N)
for(i in 1:m)
{ tetx[,i]= teta_0*df[,i] }
# defining a matrix to be filled with input and output data (for phase 2)
f.con1=matrix(ncol=N+m+s,nrow=m+s)
#defining objective and directions (phase 2)
f.obj1 = c(rep(0,N),rep(1,m),rep(1,s))
```

```
f.dir1 = c(rep("=",m+s))
for (j in 1:N) {
# defining matrix of coefficient and right hand side (phase 2)
for(i in 1:m){
f.con1[i,1:N] = c(as.numeric(df[,i])) }
for(r in (m+1):(s+m)){
f.con1[r,1:N]=c(as.numeric(df[,r])) }
f.con1[, (N+1):(N+m)]=rbind(-diag(m), matrix(0,s,m))
f.con1[, (N+m+1):(N+m+s)]=rbind(matrix(0,m,s),diag(s))
f.rhs1 = c(unlist(unname(tetx[j,])),unlist(unname(data.frame(df[j,(m+1):(m
+s)])))))
# solving model (phase 2)
results1 = lp('max', as.numeric(f.obj1), f.con1, f.dir1, f.rhs1, scale=0, com-
pute.sens=TRUE)
if (j==1) {
weights1 = results1$solution
eff1 = results1$objval
lambdas = results$duals[seq(1,N)]
} else {
weights1 = rbind(weights1, results1$solution)
eff1 = rbind(eff1, results1$objval)
lambdas = rbind(lambdas, results$duals[seq(1,N)]) } }
Es1 = data.frame(eff1, eff)
colnames(Es1) = c('phase2', 'phase1')
WriteXLS(Es1,"14-TWO-Phase-CCR.xls", row.names = T, col.names = F)
```

Example 3.14 Considering the data set given in Table 3.1 with ten DMUs each uses two inputs to produce two outputs. According to the given code for Two-phase CCR model, the results of phase 1 and phase 2 are reported in Table 3.26.

Table 3.26 Two phase results

DMU	Phase 2	Phase 1
1	207.566	1.000
2	142.975	0.762
3	64.954	0.961
4	159.347	0.625
5	184.321	1.000
6	72.694	0.440
7	112.500	0.893
8	187.321	1.000
9	117.825	0.815
10	130.138	0.202

3.7 Conclusion

In this chapter we first formulated basic DEA models and then provided R codes of these models along with numerical examples. Moreover, according to the different commands, introduced in Chap. 2, R codes of some basic DEA models have been introduced. It is evident that readers can have their own logic for writing a code. These codes have been introduced with simple examples to motivate the readers to write their own codes for more complicated models.

References

1. Charnes, A., Cooper, W.W., Rohdes, E.: Measuring the efficiency of decision making units. Eur. J. Oper. Res. **2**, 249–444 (1978)
2. Banker, R.D., Charnes, A., Cooper, W.W.: Some models for estimating technical and scale inefficiencies in data envelopment analysis. Manag. Sci. **30**(9), 1078–1092 (1984)
3. Bardhan, I., Bowlin, W.F., Cooper, W.W., Sueyoshi, T.: Models and measures for efficiency dominance in DEA, part I: additive models and med measures. J. Oper. Res. Soc. Japan **39**, 322–332 (1996)
4. Charnes, A., Cooper, W.W., Rhodes, E.: Measuring the efficiency of decision making units, short communication. Eur. J. Oper. Res. **3**, 33–39 (1979)
5. Charnes, A., Cooper, W.W., Golany, B., Seiford, L.M., Stutz, J.: Foundations of data envelopment analysis for Pareto-Koopmans efficient empirical production functions. J. Econ. (Netherlands) **30**, 91–107 (1985)

Chapter 4
Advanced DEA Models with R Codes

Abstract Using data envelopment analysis as a mathematical performance evaluation tool is much more serious for researchers and practitioners. Different data envelopment analysis models are now introduced in different fields. In addition to the classic performance evaluation models in data envelopment analysis, developed models such as super-efficiency, returns to scale, progress and regress models, and so on have been introduced in this technique that help different aspects of analytics and decision making units in performance evaluation. In this chapter, such developed DEA models are formulated, and then the corresponding R codes for these models are provided.

Keywords Data envelopment analysis · Benchmark · Ranking · Returns to scale · Malmquist productivity index · Cost efficiency · Revenue efficiency · Profit efficiency · Directional efficiency · Common weights · Congestion

4.1 Introduction

In this chapter, different developed models of DEA technique are briefly reviewed and then corresponding R codes are given with simple numerical examples.

4.2 AP Models with R Codes

In this section different formulations of AP model for ranking of DMUs are presented and then R codes for each formulation is given.

F. Hosseinzadeh Lotfi et al., *Data Envelopment Analysis with R*, Studies in Fuzziness and Soft Computing 386, https://doi.org/10.1007/978-3-030-24277-0_4

4.2.1 Input-Oriented AP Envelopment Model with R Code

Anderson and Peterson [1] introduced the AP model for super-efficiency evaluation of DMUs. As efficient DMUs are not comparable among each other furthermore, thus super-efficiency scores can help to distinguish more among the efficient units. According to the super-efficiency score it is possible to rank efficient units. Vector and matrix forms of this model are as follows.

$$\begin{aligned} \min \quad & \theta \\ \text{s.t.} \quad & -\theta x_{ip} + \sum_{\substack{j=1 \\ j\neq p}}^{n} \lambda_j x_{ij} \leq 0, \quad i = 1, \ldots, m, \\ & \sum_{\substack{j=1 \\ j\neq p}}^{n} \lambda_j y_{rj} \geq y_{rp}, \quad r = 1, \ldots, s, \\ & \lambda_j \geq 0, \quad j = 1, \ldots, n. \end{aligned} \tag{4.1}$$

$$\begin{aligned} \min \quad & [1,0,...,0]\left[\theta, \lambda_1, ..., \lambda_{p-1}, \lambda_{p+1}, ..., \lambda_n\right]^{\mathrm{t}} \\ \text{s.t.} \quad & \left[\begin{array}{c|cccc} -x_{1p} & x_{11} & \cdots & x_{1p-1}, x_{1p+1}, ..., x_{1n} \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ -x_{p} & x_{m1} & \cdots & x_{mp-1}, x_{mp+1}, ..., x_{mn} \\ \hline 0 & y_{11} & \cdots & y_{1p-1}, y_{1p+1}, ..., y_{1n} \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & y_{s1} & \cdots & y_{sp-1}, y_{sp+1}, ..., x_{sn} \end{array}\right] \left[\begin{array}{c} \theta \\ \hline \lambda_1 \\ \vdots \\ \lambda_{p-1} \\ \lambda_{p+1} \\ \vdots \\ \lambda_n \end{array}\right] \begin{array}{c} \leq_m \\ \geq_s \end{array} \left[\begin{array}{c} 0_{m\times 1} \\ \hline y_{1p} \\ \vdots \\ y_{sp} \end{array}\right] \\ & \lambda_j \geq 0, \quad j = 1, ..., n. \end{aligned} \tag{4.2}$$

Regarding what version of the R software is used, it may be necessary to use different libraries to use the write command. It should also be noted that the write. csv ( ) command can easily be used for writing. The R code of model (4.2) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df = data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
```

```
# defining a matrix to be filled with input and output data
f.con=matrix(ncol=N+1,nrow=m+s)
# number of elements
m=2
s = ncol(df)-m
N = nrow(df)
for (j in 1:N) {
# defining right hand side, directios, and objective function
f.rhs = c(rep(0,m),unlist(unname(df[j,(m+1):(m+s)])))
f.dir = c(rep("<=",m),rep(">=",s))
f.obj = c(1, rep(0,N))
# defining matrix of coefficient
for(i in 1:m){
f.con[i,1:(N+1)]=c(as.numeric(-df[j,i]),df[,i]) }
for(r in (m+1):(s+m)){
f.con[r,1:(N+1)]=c(0,as.numeric(df[,r])) }
f.con[,j+1]=c(rep(0, m+s))
# solving model
results = lp('min', as.numeric(f.obj), f.con, f.dir, f.rhs, scale=0,compute.
sens=FALSE)
if (j==1) {
weights = results$solution[1]
lambdas = results$solution[seq(2,N+1)]
} else {
weights = rbind(weights, results$solution[1])
lambdas = rbind(lambdas, results$solution[seq(2,N+1)]) } }
Es = data.frame(weights)
colnames(Es) = c('super-effi')
WriteXLS(Es,"15-AP-CCR-ENV_INP.xls", row.names = T, col.names = F)
```

Example 4.1 Consider the data set provided in Table 4.1 including ten DMUs with two inputs and two outputs are taken into consideration. The super efficiency scores are listed in Table 4.2.

Table 4.1 Data1 (input and output data)

DMU	I_1	I_2	O_1	O_2
1	20	11	8	30
2	11	40	21	20
3	32	30	34	40
4	21	30	18	50
5	20	11	6	17
6	12	43	23	58
7	7	45	28	30
8	31	45	40	20
9	19	22	27	23
10	32	11	38	45

Table 4.2 Super-efficiency scores

DMU	S.EFFI.
1	1.948
2	0.762
3	0.961
4	0.625
5	1.593
6	0.440
7	0.893
8	1.655
9	0.815
10	0.202

4.2.2 Output-Oriented AP Enveloping Model

Similar to input-oriented AP envelopment model, the vector and matrix forms of the output-oriented one are as following.

$$\begin{aligned} \max \quad & \varphi \\ \text{s.t.} \quad & \sum_{\substack{j=1 \\ j\neq p}}^{n} \lambda_j x_{ij} \leq x_{ip}, && i = 1, \ldots, m, \\ & -\varphi y_{rp} + \sum_{\substack{i=1 \\ i\neq p}}^{n} \lambda_j y_{ij} \geq 0, && r = 1, \ldots, s, \\ & \lambda_j \geq 0, && j = 1, \ldots, n. \end{aligned} \tag{4.3}$$

$$\begin{aligned} \max \quad & [1, 0, \ldots, 0]\left[\varphi, \lambda_1, \ldots, \lambda_{p-1}, \lambda_{p+1}, \ldots, \lambda_n\right]^{\mathrm{t}} \\ \text{s.t.} \quad & \left[\begin{array}{c|ccccc} 0 & x_{11} & \cdots & x_{1p-1} & x_{1p+1} & \ldots & x_{1n} \\ \vdots & \vdots & & \vdots & \vdots & & \\ 0 & x_{m1} & \cdots & x_{mp-1} & x_{mp+1} & \ldots & x_{mn} \\ \hline -y_{1p} & y_{11} & \cdots & y_{1p-1} & y_{1p+1} & \ldots & y_{1n} \\ \vdots & \vdots & & \vdots & & & \\ -y_{sp} & y_{s1} & \cdots & y_{sp-1} & y_{sp+1} & \ldots & y_{sn} \end{array}\right] \left[\begin{array}{c} \varphi \\ \hline \lambda_1 \\ \vdots \\ \lambda_{p-1} \\ \lambda_{p+1} \\ \vdots \\ \lambda_n \end{array}\right] \begin{array}{c} \leq_m \\ \geq_s \end{array} \left[\begin{array}{c} x_{1p} \\ \vdots \\ x_{mp} \\ \hline 0_{s\times 1} \end{array}\right] \\ & \lambda_j \geq 0, \quad j = 1, \ldots, n. \end{aligned} \tag{4.4}$$

The R code of model (4.4) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# defining a matrix to be filled with input and output data
f.con=matrix(ncol=N+1,nrow=m+s)
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
for (j in 1:N) {
# defining right hand side, directions, and objective
f.rhs = c(unlist(unname(df[j,(1):(m)])), rep(0,s))
f.dir = c(rep("<=",m),rep(">=",s))
f.obj = c(1, rep(0,N))
# defining matrix of coefficient
for(i in 1:m){
f.con[i,1:(N+1)]=c(0,df[,i]) }
for(r in (m+1):(s+m)){
f.con[r,1:(N+1)]=c(as.numeric(-df[j,r]),as.numeric(df[,r])) }
f.con[,j+1]=c(rep(0, m+s))
# solving mode
results = lp('max', as.numeric(f.obj), f.con, f.dir, f.rhs, scale=0,compute.
sens=TRUE)
if (j==1) {
effi = results$solution[1]
lambdas = results$solution[seq(2,N)]
} else {
effi = rbind(effi, results$solution[1])
lambdas = rbind(lambdas, results$solution[seq(2,N)]) } }
Es = data.frame(effi)
colnames(Es) = c('super-effi')
WriteXLS(Es,"16-AP-ENV-OUT.xls", row.names = F, col.names =T)
```

Example 4.2 Consider the data set in Table 4.1. The super efficiency scores considering the output-oriented AP model are listed in Table 4.3.

Table 4.3 Super-efficiency

DMU	S.EFFI.
1	0.513
2	1.313
3	1.040
4	1.600
5	0.628
6	2.274
7	1.120
8	0.604
9	1.227
10	4.960

4.2.3 Input-Oriented AP Multiplier Model

The vector and matrix forms of input-oriented AP multiplier model are as follows:

$$\begin{aligned} \max \quad & \sum_{r=1}^{s} u_r y_{rp} \\ \text{s.t.} \quad & -\sum_{i=1}^{m} v_i x_{ij} + \sum_{r=1}^{s} u_r y_{rj} \le 0, \quad i = 1, \ldots, n, \ \ j \ne p, \\ & \sum_{i=1}^{m} v_i x_{ip} = 1, \\ & v_i \ge 0, \ \ u_r \ge 0, \qquad\qquad i = 1, \ldots, m, \ \ r = 1, \ldots, s. \end{aligned} \tag{4.5}$$

$$\begin{aligned} \max \quad & \left[0, \ldots, 0, y_{1p}, \ldots, y_{sp}\right]\left[v_1, \ldots, v_m, u_1, \ldots, u_s\right]^{\mathrm{t}} \\ \text{s.t.} \quad & \left[\begin{array}{ccc:ccc} -x_{11} & \cdots & -x_{m1} & y_{11} & \cdots & y_{s1} \\ \vdots & & \vdots & \vdots & & \vdots \\ -x_{1p-1} & \cdots & -x_{mp-1} & y_{1p-1} & \cdots & y_{sp-1} \\ -x_{1p+1} & \cdots & -x_{mp-1} & y_{1p+1} & \cdots & y_{sp+1} \\ \vdots & & \vdots & \vdots & & \vdots \\ -x_{1n} & \cdots & -x_{mn} & y_{1n} & \cdots & y_{sn} \\ \hdashline x_{1p} & \cdots & x_{mp} & 0 & \cdots & 0 \end{array}\right] \left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \end{array}\right] \le_{n-1} \left[\begin{array}{c} 0_{n\times 1} \\ \hdashline 1 \end{array}\right] \\ & v_i \ge 0, \ \ u_r \ge 0, \quad i = 1, \ldots, m, \ \ r = 1, \ldots, s. \end{aligned} \tag{4.6}$$

The R code of model (4.6) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
m=2
s=ncol(df)-m
N = nrow(df)
# defining a matrix to be filled with input and output data
f.con=matrix(ncol=N,nrow=m+s-1)
# defining inputs and outputs to be used for deriving targets
inputs = data.frame(df[,1:m])
outputs = df[,(m+1):(s+m)]
# defining right hand side and directions
f.rhs = c(rep(0,N),1)
f.dir = c(rep("<=",N), "=")
for (j in 1:N) {
# defining objective and matrix of coefficient
f.obj = c(rep(0,m),as.numeric(outputs[j,]))
con = cbind(-inputs,outputs)
f.con = rbind(con,c(as.numeric(inputs[j,]), rep(0,s)))
# deleting constraint, right hand side, and direction of DMU_j
f.con = f.con[-j,]
f.rhs = c(rep(0,N),1)
f.rhs = f.rhs[-j]
f.dir = c(rep("<=",N),"=")
f.dir = f.dir[-j]
# solving model
results = lp ("max",as.numeric(f.obj), f.con, f.dir, f.rhs, scale=0, compute.
sens=TRUE)
if (j==1) {
weights = results$solution
effcrs = results$objval
lambdas = results$duals[seq(1,N)]
} else {
weights = rbind(weights, results$solution)
effcrs = rbind(effcrs, results$objval)
lambdas = rbind(lambdas, results$duals[seq(1,N)]) } }
ls = data.frame(effcrs, weights)
colnames(ls) = c('effi','v','u','u','u')
WriteXLS(ls,"17-Ap–INP-MUL.xls", row.names = F, col.names = T)
```

Table 4.4 Super-efficiency and optimal weights

DMU	S.EFFI.	v_1*	v_2*	u_1*	u_2*
1	1.948	0.026	0.022	0.000	0.025
2	0.762	0.012	0.022	0.017	0.000
3	0.961	0.009	0.016	0.013	0.000
4	0.625	0.050	0.000	0.000	0.010
5	1.593	0.056	0.000	0.019	0.000
6	0.440	0.063	0.000	0.005	0.009
7	0.893	0.071	0.000	0.000	0.014
8	1.655	0.000	0.029	0.017	0.000
9	0.815	0.021	0.007	0.011	0.000
10	0.202	0.000	0.011	0.000	0.004

Example 4.3 Consider the data set given in Table 4.1. The super efficiency scores as well as optimal weights, considering the output-oriented AP multiplier model, are listed in Table 4.4.

As "compute.sens=TRUE" thus it is also possible to obtain the optimal values of the intensifier variables.

4.2.4 Output-Oriented AP Multiplier Model

The vector and matrix forms of this output-oriented AP multiplier model are as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} v_i x_{ip} \\ \text{s.t.} \quad & -\sum_{i=1}^{m} v_i x_{ij} + \sum_{r=1}^{s} u_r y_{rj} \le 0, \quad i = 1, \ldots, n, \;\; j \ne p, \\ & \sum_{r=1}^{s} u_r y_{rp} = 1, \\ & v_i \ge 0, \;\; u_r \ge 0, \qquad\qquad i = 1, \ldots, m, \;\; r = 1, \ldots, s. \end{aligned} \tag{4.7}$$

$$
\begin{aligned}
\min \quad & \left[x_{1p},\ldots,x_{mp},0,\ldots,0\right]\left[v_1,\ldots,v_m,u_1,\ldots,u_s\right]^{\mathrm{t}} \\
\text{s.t.} \quad & \left[\begin{array}{ccc:ccc}
-x_{11} & \cdots & -x_{m1} & y_{11} & \cdots & y_{s1} \\
\vdots & & \vdots & \vdots & & \vdots \\
-x_{1p-1} & \cdots & -x_{mp-1} & y_{1p-1} & \cdots & y_{sp-1} \\
-x_{1p+1} & \cdots & -x_{mp-1} & y_{1p+1} & \cdots & y_{sp+1} \\
\vdots & & \vdots & \vdots & & \vdots \\
-x_{1n} & \cdots & -x_{mn} & y_{1n} & \cdots & y_{sn} \\
\hdashline
0 & \cdots & 0 & y_{1p} & \cdots & y_{sp}
\end{array}\right]
\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \end{array}\right]
\begin{array}{c} \leq \\ n-1 \\ = \end{array}
\left[\begin{array}{c} 0_{n-1\times 1} \\ \hdashline 1 \end{array}\right] \qquad (4.8) \\
& v_i \geq 0,\ u_r \geq 0, \quad i=1,\ldots,m,\ r=1,\ldots,s.
\end{aligned}
$$

The R code of model (4.8) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
# defining a matrix to be filled with input and output data
f.con=matrix(ncol=N,nrow=N)
# defining inputs and outputs to be used for deriving targets
inputs = data.frame(df[,1:m])
outputs = df[,(m+1):(s+m)]
# defining right hand side and directions
f.rhs = c(rep(0,N),1)
f.dir = c(rep("<=",N),"=")
for (j in 1:N) {
# defining objective and matrix of coefficient
f.obj = c(as.numeric(inputs[j,]),rep(0,s))
con = cbind(-inputs,outputs)
f.con = rbind(con,c(rep(0,m), as.numeric(outputs[j,])))
# deleting constraint, right hand side, and direction of DMU_j
f.con = f.con[-j,]
```

```
f.rhs = c(rep(0,N),1);f.rhs
f.rhs = f.rhs[-j]
f.dir = c(rep(">=",N),"=")
f.dir = f.dir[-j]
# solving model
results = lp ("min",as.numeric(f.obj), f.con, f.dir, f.rhs, scale=0, compute.
sens=F)
if (j==1) {
weights = results$solution
effcrs = results$objval
} else {
weights = rbind(weights, results$solution)
effcrs = rbind(effcrs, results$objval) } }
ls = data.frame(effcrs, weights)
colnames(ls) = c('effi','v','v','u','u')
WriteXLS(ls,"18-Ap-OUT-MUL.xls", row.names = F, col.names = T)
```

Example 4.4 Consider the data set provided in Table 4.1. The super efficiency scores as well as optimal weights are listed in Table 4.5.

As "compute.sens=F" thus it not is also possible to obtain the optimal values of the intensifier variables.

4.3 MAJ Super-Efficiency Model with R Code

Mehrabian et al. [2] presented a DEA model for more discriminating among efficient DMUs. The vector-based and matrix-based forms of this model are as follows:

Table 4.5 Super-efficiency and optimal weights for Example 4.4

DMU	S.EFFI.	v_1*	v_2*	u_1*	u_2*
1	0.513	0.014	0.012	0.000	0.013
2	1.313	0.016	0.029	0.023	0.000
3	1.040	0.010	0.017	0.013	0.000
4	1.600	0.080	0.000	0.000	0.015
5	0.628	0.035	0.000	0.012	0.000
6	2.274	0.142	0.000	0.011	0.021
7	1.120	0.080	0.000	0.000	0.015
8	0.604	0.000	0.017	0.010	0.000
9	1.227	0.026	0.009	0.014	0.000
10	4.960	0.000	0.055	0.000	0.019

$$
\begin{aligned}
\min \quad & 1 + w_p^+ - w_p^- \\
\text{s.t.} \quad & \sum_{\substack{j=1 \\ j \neq p}}^{n} \lambda_j x_{ij} - w_p^+ + w_p^- \leq x_{ip}, \quad i = 1, \ldots, m, \\
& \sum_{\substack{j=1 \\ j \neq p}}^{n} \lambda_j y_{rj} \geq y_{rp}, \qquad r = 1, \ldots, s, \\
& \lambda_j \geq 0, \qquad j = 1, \ldots, n, \ \ j \neq p.
\end{aligned}
\tag{4.9}
$$

$$
1 + (\min \left[\underbrace{0, \ldots, 0}_{n-1}, 1, -1 \right] \left[\lambda_1, \ldots, \lambda_{p-1}, \lambda_{p+1}, \ldots, \lambda_n, w_p^+, w_p^- \right]^{\mathrm{t}})
$$

$$
\text{s.t.} \quad \left[\begin{array}{ccccccc:cc}
x_{11} & \cdots & x_{1p-1} & x_{1p+1} & \cdots & x_1 & 1 & -1 \\
\vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots \\
x_{m1} & \cdots & x_{mp-1} & x_{mp+1} & \cdots & x_{mn} & 1 & -1 \\
\hdashline
y_{11} & \cdots & y_{1p-1} & y_{1p+1} & \cdots & y_{1n} & 0 & 0 \\
\vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots \\
y_{s1} & \cdots & y_{sp-1} & y_{sp+1} & \cdots & y_{sn} & 0 & 0
\end{array}\right]
\left[\begin{array}{c}
\lambda_1 \\ \vdots \\ \lambda_{p-1} \\ \lambda_{p+1} \\ \vdots \\ \lambda_n \\ \hdashline w_p^+ \\ \vdots \\ w_p^-
\end{array}\right]
\begin{array}{c} \leq_m \\ \geq_s \end{array}
\left[\begin{array}{c}
x_{1p} \\ \vdots \\ x_{mp} \\ \hdashline y_{1p} \\ \vdots \\ y_{sp}
\end{array}\right]
$$

$$
\lambda_j \geq 0, \quad j = 1, \ldots, n, \ j \neq p.
\tag{4.10}
$$

The R code of model (4.10) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
```

```
df = data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# defining a matrix to be filled with inputs and output data
f.con=matrix(ncol=N+2,nrow=m+s)
# number of elements
m=2
s = ncol(df)-m
N = nrow(df)
for (j in 1:N) {
# defining right hand side, directions, and objective
f.rhs = c(as.numeric(df[j,1:m]), as.numeric(df[j,(m+1):(m+s)]))
f.dir = c(rep("<=",m),rep(">=",s))
f.obj = c(rep(0,N),1,-1)
# defining matrix of coefficient
for(i in 1:m){
f.con[i,1:N]=c(df[,i])
}
for(r in (m+1):(s+m)){
f.con[r,1:N]=c(df[,r])
}
f.con[,(N+1)]=c(rep(-1,m),rep(0,s))
f.con[,(N+2)]=c(rep(1,m),rep(0,s))
f.con[,j]=c(rep(0, m+s))
# solving model
result = lp('min', as.numeric(f.obj), f.con, f.dir, f.rhs, scale=0,compute.
sens=FALSE)
w=result$objval
if (j==1) {
MAJ = c(result$objval, (1+w))
} else {
MAJ = rbind(MAJ, c(result$objval, (1+w))) } }
EFFI = data.frame(MAJ)
colnames(EFFI) = c('W*', 'S.EFFI-Maj')
WriteXLS(EFFI,"19-MAJ-SUP_EFF.xls", row.names = F, col.names = T)
```

Table 4.6 Super-efficiency for Example 4.5

DMU	W*	S.EFFI.MAJ
1	19.462	20.462
2	−6.944	−5.944
3	−1.519	−0.519
4	−7.500	−6.500
5	10.667	11.667
6	−8.964	−7.964
7	−1.500	−0.500
8	22.914	23.914
9	−6.634	−5.634
10	−61.808	−60.808

Example 4.5 Consider the data set provided in Table 4.1. The super efficiency scores as well as optimal weights are listed in Table 4.6.

4.4 Norm L_1 Super-Efficiency Model with R Code

Another super-efficiency DEA model has been presented by Jahanshahloo et al. [3]. The vector and matrix forms of this model are as follows:

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{m} x_i - \sum_{r=1}^{s} y_r + \alpha \\
\text{s.t.} \quad & \sum_{\substack{j=1 \\ j\neq p}}^{n} \lambda_j x_{ij} - x_i \leq 0, \quad i = 1, \ldots, m, \\
& \sum_{\substack{j=1 \\ j\neq p}}^{n} \lambda_j y_{rj} - y_r \geq 0, \quad r = 1, \ldots, s, \\
& x_i \geq x_{ip}, \quad i = 1, \ldots, m, \\
& y_r \geq y_{rp}, \quad r = 1, \ldots, s, \\
& \lambda_j \geq 0, \quad j = 1, \ldots, n, \; j \neq p, \\
& y_r \geq 0, \quad r = 1, \ldots, s, \\
& x_i \geq 0, \quad i = 1, \ldots, m.
\end{aligned}
\tag{4.11}
$$

$$\alpha + (\max \quad \underset{1\times(n-1)}{0} \ , \underset{1\times m}{1} \ , \underset{1\times s}{-1} \Big[\lambda_1,...,\lambda_{p-1},\lambda_{p+1},...,\lambda_n,x_1,...,x_m,y_1,...,y_s\Big]^{\mathrm{t}})$$

s.t.

$$\left[\begin{array}{ccccccc|c|c} x_{11} & \cdots & x_{1p-1} & x_{1p+1} & \cdots & x_{1n} & & & \\ \vdots & & \vdots & \vdots & & \vdots & & & \\ x_{m1} & \cdots & x_{mp-1} & x_{mp+1} & \cdots & x_{mn} & -I_m & o_{m\times s} & \\ \hline y_{11} & \cdots & y_{1p-1} & y_{1p+1} & \cdots & x_{1n} & & & \\ \vdots & & \vdots & \vdots & & \vdots & o_{s\times m} & -I_s & \\ y_{s1} & \cdots & y_{sp-1} & x_{p+1} & \cdots & x_{sn} & & & \end{array}\right] \left[\begin{array}{c} \lambda_1 \\ \vdots \\ \lambda_{p-1} \\ \lambda_{p+1} \\ \vdots \\ \lambda_n \\ \hline x_1 \\ \vdots \\ x_m \\ \hline y_1 \\ \vdots \\ y_s \end{array}\right] \begin{array}{c} \geq_s \\ \geq_m \\ \leq_s \end{array} \left[\begin{array}{c} 0_{m\times 1} \\ \hline 0_{s\times 1} \\ \hline x_{1p} \\ \vdots \\ x_{mp} \\ \hline y_{1p} \\ \vdots \\ x_{sp} \end{array}\right] \tag{4.12}$$

$$\lambda_j \geq 0, \quad j = 1,...,n, \ \ j \neq p,$$
$$y_r \geq 0, \quad r = 1,...,s,$$
$$x_i \geq 0, \quad i = 1,...,m,$$

where $\alpha = \sum_{r=1}^{s} y_{rp} - \sum_{i=1}^{m} x_{ip}$.

The R code of model (4.12) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df = data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# defining a matrix to be filled with inputs and outputs data
f.con=matrix(ncol=N+2,nrow=m+s)
# number of elements
m=2
s = ncol(df)-m
N = nrow(df)
```

```
for (j in 1:N) {
# defining right hand side, directions, and objective
f.rhs = c(as.numeric(df[j,1:m]), as.numeric(df[j,(m+1):(m+s)]))
f.dir = c(rep("<=",m),rep(">=",s))
f.obj = c(rep(0,N),1,-1)
# defining matrix of coefficient
for(i in 1:m){
f.con[i,1:N]=c(df[,i])
}
for(r in (m+1):(s+m)){
f.con[r,1:N]=c(df[,r])
}
f.con[,(N+1)]=c(rep(-1,m),rep(0,s))
f.con[,(N+2)]=c(rep(1,m),rep(0,s))
f.con[,j]=c(rep(0, m+s))
# solving model
result = lp('min', as.numeric(f.obj), f.con, f.dir, f.rhs, scale=0, compute.
sens=FALSE)
w=result$objval
if (j==1) {
NL = c(result$objval, (1+w))
} else {
NL = rbind(NL, c(result$objval, (1+w))) } }
EFFI = data.frame(NL)
colnames(EFFI) = c('super-effi-NL');EFFI
WriteXLS(EFFI,"20-MAJ-SUP_EFF.xls", row.names = FALSE, col.names
= FALSE)
```

Example 4.6 Consider the data set provided in Table 4.1. The super efficiency scores are listed in Table 4.7.

Table 4.7 Optimal objective and super efficiency for Example 4.6

DMU	OP.OBJ	SUPER-EFFI-NL
1	19.462	20.462
2	−6.944	−5.944
3	−1.519	−0.519
4	−7.500	−6.500
5	10.667	11.667
6	−8.964	−7.964
7	−1.500	−0.500
8	22.914	23.914
9	−6.634	−5.634
10	−61.808	−60.808

As "compute.sens=TRUE" thus it is also possible to obtain the optimal values of dual variables.

4.5 Returns to Scale—CCR Models with R Codes

In this section returns to scale DEA models are presented and the R codes are provided for all formulated models.

4.5.1 Returns to Scale—CCR Envelopment Model with R Code

Banker et al. [4] utilizing CCR model distinguished the type of returns to scale of DMUs. The vector-based and matrix-based forms of this model are as follows:

$$\begin{aligned}
\max \quad & \sum_{r=1}^{s} u_r y_{rP} + u_0^+ - u_0^- \\
\text{s.t.} \quad & -\sum_{i=1}^{m} v_i x_{ij} + \sum_{r=1}^{s} u_r y_{rj} + u_0^+ - u_0^- \le 0, \quad j = 1, \ldots, n, \\
& \sum_{i=1}^{m} v_i x_{ip} = 1, \\
& u_r \ge 0, \quad v_i \ge 0, \qquad\qquad r = 1, \ldots, s, \quad i = 1, \ldots, m, \\
& u_0^+ \ge 0, \quad u_0^- \ge 0.
\end{aligned} \tag{4.13}$$

$$\begin{aligned}
\min(\max) \quad & \sum_{j=1}^{n} \lambda_j \\
\text{s.t.} \quad & -\theta x_{ip} + \sum_{j=1}^{n} \lambda_j x_{ij} \le 0, \quad i = 1, \ldots, m, \\
& \sum_{j=1}^{n} \lambda_j y_{rj} \ge y_{rp}, \qquad r = 1, \ldots, s, \\
& \lambda_j \ge 0, \qquad\qquad j = 1, \ldots, n.
\end{aligned} \tag{4.14}$$

$$
\begin{aligned}
\max \quad & \left[0,\ldots,0,y_{1p},\ldots,y_{sp},1,-1\right]\left[v_1,\ldots,v_m,u_1,\ldots,u_s,u_0^+,u_0^-\right]^{\mathrm{t}} \\
\text{s.t.} \quad & \left[\begin{array}{ccc:ccc:cc}
-x_{11} & \cdots & -x_{m1} & y_{11} & \cdots & y_{s1} & 1 & -1 \\
\vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots \\
-x_{1n} & \cdots & -x_{mn} & y_{1n} & \cdots & y_{sn} & 1 & -1 \\
\hdashline
x_{1p} & \cdots & x_{mp} & 0 & \cdots & 0 & 0 & 0
\end{array}\right]
\left[\begin{array}{c}
v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \\ \hdashline u_0^+ \\ u_0^-
\end{array}\right]
=_{n+1}
\left[\begin{array}{c}
0_{n\times 1} \\ \hdashline 1
\end{array}\right] \qquad (4.15) \\
& u_r \geq 0,\ v_i \geq 0, \quad r=1,\ldots,s,\ i=1,\ldots,m, \\
& u_0^+ \geq 0,\ u_0^- \geq 0.
\end{aligned}
$$

$$
\begin{aligned}
\min(\max) \quad & \left[0,1,\ldots,1\right]\left[\theta,\lambda_1,\ldots,\lambda_n\right]^{\mathrm{t}} \\
\text{s.t.} \quad & \left[\begin{array}{c:ccc}
-x_{1p} & x_{11} & \cdots & x_{1n} \\
\vdots & \vdots & & \vdots \\
-x_{mp} & x_{m1} & \cdots & x_{mn} \\
\hdashline
0 & y_{11} & \cdots & y_{1n} \\
\vdots & \vdots & & \vdots \\
0 & y_{s1} & \cdots & y_{sn}
\end{array}\right]
\left[\begin{array}{c}
\theta \\ \hdashline \lambda_1 \\ \vdots \\ \lambda_n
\end{array}\right]
\begin{array}{c}
\leq_m \\ \hdashline \geq_s
\end{array}
\left[\begin{array}{c}
0_{m\times 1} \\ \hdashline y_{1p} \\ \vdots \\ y_{sp}
\end{array}\right] \qquad (4.16) \\
& \lambda_j \geq 0, \quad j=1,\ldots,n.
\end{aligned}
$$

Having optimal solution of BCC model, for the BCC efficient DMUs the following cases can be considered:

If $\theta^*_{CCR}=1$ and $\sum_{j=1}^{u}\lambda_i^*=1$ in all optimal solutions, then DMU under evaluation has constant returns to scale form of the technology.
If $\theta^*_{CCR}<1$ and $\sum_{j=1}^{u}\lambda_i^*>1$ in all optimal solutions, then DMU under evaluation has non increasing returns to scale form of the technology.
If $\theta^*_{CCR}>1$ and $\sum_{j=1}^{u}\lambda_i^*<1$ in all optimal solutions, then DMU under evaluation has non decreasing returns to scale form of the technology.

The R code of models (4.14) and (4.15) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
# defining directions, right hand side, and constraints
f.rhs0 = c(rep(0,N),1)
f.dir0 = c(rep("<=",N),"=")
con01 = cbind(-df[,1:m],df[,(m+1):(m+s)], 1, -1)
for (j in 1:N) {
f.obj0 = c(rep(0,m),as.numeric(df[j,(m+1):(m+s)]), 1, -1)
con02= c(as.numeric(df[j,1:m]), rep(0,s),0,0)
f.con0 = rbind(con01,con02)
# solving model
resultss0 = lp ("max",f.obj0, f.con0, f.dir0, f.rhs0, scale=0, compute.
sens=TRUE)
multipliersss0 = resultss0$solution
u0 = multipliersss0[s+m+1]-multipliersss0[s+m+2]
if (j==1) {
efbcc=resultss0$objval
weightss0 = u0
lambdass0= resultss0$duals[seq(1,N+1)]
} else {
ef= rbind(efbcc, resultss0$objval)
weightss0 = rbind(weightss0,u0)
if ((resultss0$objval) == 1) {effvr0 = rbind(effvr0, resultss0$objval)} else
{effvr0 = rbind(effvr0, 'bcc-ineffi')}
lambdass0 = rbind(lambdass0,resultss0$duals[seq(1,N+1)])
}
}
f.con=matrix(ncol=N+1,nrow=m+s)
m=2
s = ncol(df)-m
N = nrow(df)
for (j in 1:N) {
```

```
f.rhs = c(rep(0,m),df[j,(m+1):(m+s)])
f.dir = c(rep("<=",m),rep(">=",s))
f.obj = c(1, rep(0,N))
for(i in 1:m){
f.con[i,1:(N+1)] = c(as.numeric(-df[j,i]),df[,i])
}
for(r in (m+1):(s+m)) {
f.con[r,1:(N+1)] = c(0,as.numeric(df[,r]))
}
results = lp('min', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
efccr = results$solution[1]
lambdas = results$solution[seq(2,N+1)]
sumlan=lambdas %*% c(rep(1,N));sumlan
} else {
efccr = rbind(efccr, results$solution[1])
lambdas = rbind(lambdas, results$solution[seq(2,N+1)])
sumlan=lambdas %*% c(rep(1,N))
}
}
Es = data.frame(ef, efccr, weightss0, sumlan)
colnames(Es) = c('effi bcc', 'effi ccr', 'u0', 'sum-lambda'); Es
WriteXLS(Es,"21-RTS-env-ccr.xls", row.names = TRUE, col.names =
TRUE)
```

Example 4.7 Consider the data set provided in Table 4.1. The optimal solutions are listed in Table 4.8. In this table, efficiency of BCC and CCR models along with minimum and maximum sum of lambdas are reported.

Table 4.8 Optimal values

DMU	BCC. EFFI.	CCR. EFFI.	MIN. SUM	MAX. SUM
1	1.000	1.000	1.000	1.000
2	0.297	0.762	0.762	0.297
3	1.000	0.961	0.961	1.000
4	0.297	0.625	0.625	0.297
5	1.000	1.000	1.000	1.000
6	0.297	0.440	0.440	0.297
7	1.000	0.893	0.893	1.000
8	0.297	1.000	1.000	0.297
9	1.000	0.815	0.815	1.000
10	0.297	0.202	0.202	0.297

4.5.2 Returns to Scale—DEA Multiplier Model with R Code

According to the BCC multiplier model, Banker et al. [4] distinguished the type of return to scale of DMU_p under evaluation. Vector and matrix forms of these models are as follows.

$$
\begin{aligned}
\max \quad & \sum_{r=1}^{s} u_r y_{rP} + u_0^+ - u_0^- \\
\text{s.t.} \quad & -\sum_{i=1}^{m} v_i x_{ij} + \sum_{r=1}^{s} u_r y_{rP} + u_0^+ - u_0^- \le 0, \quad j = 1, \ldots, n, \\
& \sum_{i=1}^{m} v_i x_{ip} = 1, \\
& u_r \ge 0, \quad v_i \ge 0, \qquad\qquad r = 1, \ldots, s, \quad i = 1, \ldots, m, \\
& u_0^+ \ge 0, \quad u_0^- \ge 0.
\end{aligned}
\tag{4.17}
$$

$$
\max \left[0, \ldots, 0, y_{1p}, \ldots, y_{sp}, 1, -1\right] \left[v_1, \ldots, v_m, u_1, \ldots, u_s, u_0^+, u_0^-\right]^{\mathrm{t}}
$$

$$
\text{s.t.} \quad \left[\begin{array}{ccc|ccc|cc}
-x_{11} & \cdots & -x_{m1} & y_{11} & \cdots & y_{s1} & 1 & -1 \\
\vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots \\
-x_{1n} & \cdots & -x_{mn} & y_{1n} & \cdots & y_{sn} & 1 & -1 \\
\hline
x_{1p} & \cdots & x_{mp} & 0 & \cdots & 0 & 0 & 0
\end{array}\right]
\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hline u_1 \\ \vdots \\ u_s \\ \hline u_0^+ \\ u_0^- \end{array}\right]
\begin{array}{l} \le_n \\ =_1 \end{array}
\left[\begin{array}{c} 0_{n\times 1} \\ \hline 1 \end{array}\right]
\tag{4.18}
$$

$$
\begin{aligned}
& u_r \ge 0, \; v_i \ge 0, \quad r = 1, \ldots, s, \; i = 1, \ldots, m, \\
& u_0^+ \ge 0, \; u_0^- \ge 0.
\end{aligned}
$$

$$
\begin{aligned}
\max \quad & u_0^+ - u_0^- \\
\text{s.t.} \quad & -\sum_{i=1}^{m} v_i x_{ij} + \sum_{r=1}^{s} u_r y_{rj} + u_0^+ - u_0^- \le 0, \quad j = 1, \ldots, n, \\
& \sum_{r=1}^{s} u_r y_{rp} + u_0^+ - u_0^- = 1, \\
& \sum_{i=1}^{m} v_i x_{ip} = 1, \\
& u_r \ge 0, \quad v_i \ge 0, \qquad\qquad r = 1, \ldots, s, \quad i = 1, \ldots m, \\
& u_0^+ \ge 0, \quad u_0^- \ge 0.
\end{aligned}
\tag{4.19}
$$

$$
\begin{aligned}
\min \quad & u_0^+ - u_0^- \\
\text{s.t.} \quad & -\sum_{i=1}^{m} v_i x_{ij} + \sum_{r=1}^{s} u_r y_{rj} + u_0^+ - u_0^- \le 0, \quad j = 1, \ldots, n, \\
& \sum_{r=1}^{s} u_r y_{rp} + u_0^+ - u_0^- = 1, \\
& \sum_{i=1}^{m} v_i x_{ip} = 1, \\
& u_r \ge 0, \quad v_i \ge 0, \qquad\qquad r = 1, \ldots, s, \quad i = 1, \ldots m, \\
& u_0^+ \ge 0, \quad u_0^- \ge 0.
\end{aligned}
\tag{4.20}
$$

$$
\begin{aligned}
&\max \quad \left[0,\ldots,0,0,\ldots,0,1,-1\right]\left[v_1,\ldots,v_m,u_1,\ldots,u_s,u_0^+,u_0^-\right]^t \\
&\text{s.t.} \\
&\begin{bmatrix}
-x_{11} & \cdots & -x_{m1} & | & y_{11} & \cdots & y_{s1} & | & 1 & -1 \\
\vdots & \ddots & \vdots & | & \vdots & \ddots & \vdots & | & \vdots & \vdots \\
-x_{1n} & \cdots & -x_{mn} & | & y_{1n} & \cdots & y_{sn} & | & 1 & -1 \\
- & - & - & | & - & - & - & | & - & - \\
0 & \cdots & 0 & | & y_{1p} & \cdots & y_{sp} & | & 1 & -1 \\
- & - & - & | & - & - & - & | & - & - \\
x_{1p} & \cdots & x_{mp} & | & 0 & \cdots & 0 & | & 0 & 0
\end{bmatrix}
\begin{bmatrix} v_1 \\ \vdots \\ v_m \\ \hline u_1 \\ \vdots \\ u_s \\ \hline u_0^+ \\ u_0^- \end{bmatrix}
\begin{matrix} \le_n \\ \\ \\ \\ =_1 \\ \\ =_1 \\ \\ \\ \end{matrix}
\begin{bmatrix} 0_{n\times 1} \\ \\ \\ \\ 1 \\ \\ 1 \\ \\ \\ \end{bmatrix} \\
&u_r \ge 0,\ v_i \ge 0, u_0^+ \ge 0,\ u_0^- \ge 0, \qquad r = 1,\ldots,s, i = 1,\ldots m.
\end{aligned}
\tag{4.21}
$$

$$
\begin{aligned}
&\min \quad \left[0,\ldots,0,0,\ldots,0,1,-1\right]\left[v_1,\ldots,v_m,u_1,\ldots,u_s,u_0^+,u_0^-\right]^t \\
&\text{s.t.} \\
&\begin{bmatrix}
-x_{11} & \cdots & -x_{m1} & | & y_{11} & \cdots & y_{s1} & | & 1 & -1 \\
\vdots & \ddots & \vdots & | & \vdots & \ddots & \vdots & | & \vdots & \vdots \\
-x_{1n} & \cdots & -x_{mn} & | & y_{1n} & \cdots & y_{sn} & | & 1 & -1 \\
- & - & - & | & - & - & - & | & - & - \\
0 & \cdots & 0 & | & y_{1p} & \cdots & y_{sp} & | & 1 & -1 \\
- & - & - & | & - & - & - & | & - & - \\
x_{1p} & \cdots & x_{mp} & | & 0 & \cdots & 0 & | & 0 & 0
\end{bmatrix}
\begin{bmatrix} v_1 \\ \vdots \\ v_m \\ \hline u_1 \\ \vdots \\ u_s \\ \hline u_0^+ \\ u_0^- \end{bmatrix}
\begin{matrix} \le_n \\ \\ \\ \\ =_1 \\ \\ =_1 \\ \\ \\ \end{matrix}
\begin{bmatrix} 0_{n\times 1} \\ \\ \\ \\ 1 \\ \\ 1 \\ \\ \\ \end{bmatrix} \\
&u_r \ge 0,\ v_i \ge 0,\ u_0^+ \ge 0,\ u_0^- \ge 0, \quad r = 1,\ldots,s,\ i = 1,\ldots m.
\end{aligned}
\tag{4.22}
$$

If $u_0^+ < 0$ then DMU_p has increasing returns to scale and if $u_0^+ = 0$ then it has constant returns to scale. In the case that $u_0^+ > 0$ when $u_0^- > 0$ returns to scale is increasing and if $u_0^- \leq 0$ it is decreasing or constant.

The R code of models (4.19)–(4.22) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
# defining right hand side and directions
f.rhs0 = c(rep(0,N),1)
f.dir0 = c(rep("<=",N),"=")
for (j in 1:N) {
# defining objective and matrix of coefficient
f.obj0 = c(rep(0,m),as.numeric(df[j,(m+1):(m+s)]), 1, -1)
con01 = cbind(-df[,1:m],df[,(m+1):(m+s)], 1, -1)
con02= c(as.numeric(df[j,1:m]), rep(0,s),0,0)
f.con0 = rbind(con01,con02)
# solving model
resultss0 = lp ("max",f.obj0, f.con0, f.dir0, f.rhs0, scale=0, compute.
sens=TRUE)
multipliersss0 = resultss0$solution
u10 = multipliersss0[s+m+1]-multipliersss0[s+m+2]
if (j==1) {
weightss0 = u10
effvrss0 = resultss0$objval
lambdass0= resultss0$duals[seq(1,N+1)]
} else {
weightss0 = rbind(weightss0,u10)
effvrss0 = rbind(effvrss0, resultss0$objval)
lambdass0 = rbind(lambdass0,resultss0$duals[seq(1,N+1)]) } }
# defining right hand side and directions
f.rhs = c(rep(0,N),1,1)
f.dir = c(rep("<=",N),"=", "=")f.obj = c(rep(0,m), rep(0,s), 1, -1)
for (j in 1:N){
# defining matrix of coefficient
```

```
con1 = cbind(-df[,1:m],df[,(m+1):(m+s)], 1, -1)
con2 = c(rep(0,m), as.numeric(df[j,(m+1):(m+s)]), 1,-1)
con3 = c(as.numeric(df[j,(1):(m)]), rep(0,s), 0,0)
f.con = rbind(con1, con2, con3)
# solving model
result = lp ("max",f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=TRUE)
multiplier = result$solution
u0 = multiplier[s+m+1]-multiplier[s+m+2]
if (j==1) {
weight = u0
eff = result$objval
} else {
weight = rbind(weight,u0)
eff = rbind(eff,result$objval)
}
}
# defining right hand side, directions, and objective
f.rhs1 = c(rep(0,N),1,1)
f.dir1 = c(rep("<=",N),"=", "=")
f.obj1 = c(rep(0,m), rep(0,s), 1, -1)
for (j in 1:N) {
# defining matrix of coefficient
con4 = cbind(-df[,1:m],df[,(m+1):(m+s)], 1, -1)
con5 = c(rep(0,m), as.numeric(df[j,(m+1):(m+s)]), 1,-1)
con6 = c(as.numeric(df[j,(1):(m)]), rep(0,s), 0,0)
f.con1 = rbind(con4, con5, con6)
# solving model
result1 = lp ("min",f.obj1, f.con1, f.dir1, f.rhs1, scale=0, compute.
sens=TRUE)
multiplier1 = result1$solution
u00 = multiplier1[s+m+1]-multiplier1[s+m+2]
if (j==1) {
weight1 = u00
eff1 = result1$objval
} else {
weight1 = rbind(weight1,u00)
eff1 = rbind(eff1,result1$objval) } }
lsrts = data.frame(effvrss0, weightss0, weight1, weight)
colnames(lsrts) = c("effi-BCC", 'u0-BCC', 'Min-u0', 'Max-u0')
WriteXLS(ls,"22-RTS-tow-models-MUL.xls", row.names = F,
col.names = T)
```

Table 4.9 Optimal solutions

DMU	BCC.EFFI.	U_0*	MIN U_0*	MAX U_0*
1	1.000	0.000	0.000	1.000
2	0.844	0.844	0.000	0.000
3	1.000	−0.072	−0.836	−0.072
4	0.731	0.643	0.000	0.000
5	1.000	0.000	−426.500	0.650
6	0.875	0.804	0.000	0.000
7	1.000	0.643	0.643	1.000
8	1.000	0.000	0.000	0.577
9	0.851	−0.127	0.000	0.000
10	0.297	0.297	0.000	0.000

Example 4.8 Consider the data set provided in Table 4.1. The optimal solutions are listed in Table 4.9. In this table, efficiency of BCC model and optimal values of u_0 and maximum and minimum values of u_0 are given.

4.6 Cost Efficiency Model with R Code

Camanho and Dyson [5] introduced cost efficiency DEA model. The vector and matrix forms of this model are formulated as follows:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m} c_i x_i \\
\text{s.t.} \quad & \sum_{j=1}^{n} \lambda_j x_{ij} - x_i = 0, \quad i = 1, \ldots, m, \\
& \sum_{j=1}^{n} \lambda_j y_{rj} \geq y_{rp}, \quad r = 1, \ldots, s, \\
& \lambda_j \geq 0, \quad i = 1, \ldots, n.
\end{aligned} \tag{4.23}$$

$$\begin{aligned}
\min \quad & [0, \ldots, 0, c_1, \ldots, c_m] [\lambda_1, \ldots, \lambda_n, x_1, \ldots, x_m]^{\mathrm{t}} \\
\text{s.t.} \quad & \left[\begin{array}{ccc:c} x_{11} & \cdots & x_{1n} & \\ \vdots & & \vdots & -I_m \\ x_{m1} & \cdots & x_{mn} & \\ \hdashline y_{11} & \cdots & y_{1n} & \\ \vdots & & \vdots & o_{s\times m} \\ y_{s1} & \cdots & y_{sn} & \end{array}\right] \left[\begin{array}{c} \lambda_1 \\ \vdots \\ \lambda_n \\ \hdashline x_1 \\ \vdots \\ x_m \end{array}\right] \begin{array}{c} =_m \\ \hdashline \geq_s \end{array} \left[\begin{array}{c} 0_{s\times 1} \\ \hdashline y_{1p} \\ \vdots \\ y_{sp} \end{array}\right] \\
& \lambda_j \geq 0, \quad i = 1, \ldots, n.
\end{aligned} \tag{4.24}$$

In this model c is the vector of prices. Having the optimal solution of this model finally the cost efficiency score of DMU_p can be obtained as $CE_P = c^t x^* / c^t x_p$.

The R code of model (4.24) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df = data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# number of elements
m=2
s = ncol(df)-m
N = nrow(df)
# defining matrixes
f.con=matrix(ncol=N,nrow=m+s)
rcost=matrix(ncol=m,nrow=N)
costt=matrix(ncol=1,nrow=N)
# vector of costs
A=c(15, 21)
for (j in 1:N) {
# defining right hand side, directions, and objective
f.rhs = c(rep(0,m),unlist(unname(df[j,(m+1):(m+s)])))
f.dir = c(rep("=",m),rep(">=",s))
f.obj = c(rep(0,N), A)
for(i in 1:m){
f.con[i,1:N]=c(df[,i])
rcost[j,i]=A[i]*df[j,i]
costt[,1]=rcost[,1]+rcost[,2] }
for(r in (m+1):(s+m)){
f.con[r,1:N]=c(df[,r]) }
c1=rbind(-diag(m), matrix(0,s,m))
final.con = cbind(f.con, c1)
# solving model
results = lp('min', as.numeric(f.obj), final.con, f.dir, f.rhs, scale=0, compute.
sens=FALSE)
if (j==1) {
weights = results$solution[(N+1):(N+2)]
mincost = results$objval
} else {
weights = rbind(weights, results$solution[(N+1):(N+2)])
mincost = rbind(mincost, results$objval) } }
```

```
ccost=costt
ce=mincost/ccost
COEE=data.frame(weights, mincost, ccost, ce)
colnames(COEE) = c('x', 'x', 'mincost', 'cost', 'costeffi')
WriteXLS(COE,"23-COST-EFFI.xls", row.names = F, col.names = T)
```

Example 4.9 Consider the data set provided in Table 4.1. The optimal cost efficiency scores are listed in Table 4.10. In this table, optimal input quantities along with current and minimum cost are gathered.

4.7 Revenue Efficiency DEA Model with R Code

Wang et al. [6] introduced a model to obtain the revenue efficiency score of DMU_p under evaluation. The vector and matrix forms of this model can be stated as follows:

$$\begin{aligned} \max \quad & \sum_{r=1}^{s} w_r y_r \\ \text{s.t.} \quad & \sum_{j=1}^{n} \lambda_j x_{ij} \le x_{ip}, \qquad i = 1, \ldots, m, \\ & \sum_{j=1}^{n} \lambda_j y_{rj} - y_r = 0, \quad r = 1, \ldots, s, \\ & \lambda_j \ge 0, \qquad j = 1, \ldots, n. \end{aligned} \tag{4.25}$$

Table 4.10 Cost efficiency scores and optimal solutions

DMU	X_1*	X_2*	MIN.COST	CURRENT COST	COST.EFFI.
1	15.000	27.000	792.000	792.000	1.000
2	14.667	26.400	774.400	1032.000	0.750
3	25.000	45.000	1320.000	1389.000	0.950
4	12.500	22.500	660.000	1497.000	0.441
5	28.667	51.600	1513.600	2013.000	0.752
6	8.000	14.400	422.400	1962.000	0.215
7	12.500	22.500	660.000	1386.000	0.476
8	32.667	58.800	1724.800	1860.000	0.927
9	24.333	43.800	1284.800	1803.000	0.713
10	10.192	18.346	538.154	2991.000	0.180

$$\begin{aligned}
\max \quad & [0,...,0,w_1,...,w_s]\left[\lambda_1,...,\lambda_n,y_1,...,y_s\right]^{\mathrm{t}} \\
\text{s.t.} \quad & \left[\begin{array}{ccc:c} x_{11} & \cdots & x_{1p-1} & \\ \vdots & & \vdots & o_{m\times s} \\ x_{m1} & \cdots & x_{mp-1} & \\ \hdashline y_{11} & \cdots & y_{1n} & \\ \vdots & & \vdots & -I_s \\ y_{s1} & \cdots & y_{sn} & \end{array}\right]\left[\begin{array}{c} \lambda_1 \\ \vdots \\ \lambda_n \\ \hdashline y_1 \\ \vdots \\ y_s \end{array}\right] \begin{array}{c} \leq_m \\ \hdashline =_s \end{array} \left[\begin{array}{c} x_{1p} \\ \vdots \\ x_{mp} \\ \hdashline 0_{s\times 1} \end{array}\right] \\
& \lambda_j \geq 0, \quad i=1,...,n.
\end{aligned} \tag{4.26}$$

where w is the vector of output prices. Finally, revenue efficiency of the DMU_p under evaluation will be obtained as $RE_P = w^t y_p / w^t y^*$.

The R code of model (4.26) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df = data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# required libraries
m=2
s = ncol(df)-m
N = nrow(df)
# defining matrices
f.con=matrix(ncol=N,nrow=m+s)
revenue=matrix(ncol=m+s,nrow=N)
rev=matrix(ncol=1,nrow=N)
# vector of prices
w=c(0,0,15,18)
for (j in 1:N) {
# defining right hand side, directions, and objective function
f.rhs = c(unlist(unname(df[j,1:m])),rep(0,s))
f.dir = c(rep("<=",m),rep("=",s))
f.obj = c(rep(0,N), 15, 18)
# filling inputs and outputs in matrices
for(i in 1:m){
f.con[i,1:N]=c(df[,i]) }
for(r in (m+1):(s+m)){
f.con[r,1:N]=c(df[,r]) revenue[j,1]=0
```

```
revenue[j,2]=0
revenue[j,r]= w[r] * df[j,r]
rev[j,1]=revenue[j,3]+revenue[j,4] }
c1=rbind(matrix(0,m,s), -diag(s))
final.con = cbind(f.con, c1)
# solving model
results = lp('max', as.numeric(f.obj), final.con, f.dir, f.rhs, scale=0,
compute.sens=FALSE)
if (j==1) {
maxrev = results$objval
a=maxrev
b=rev[j,1]
ce= b/a
weights = results$solution[N+1]
} else {
maxrev = rbind(maxrev, results$objval)
a=maxrev
b=rev[j,1]
c=b/a
ce=rbind(ce, c)
weights = rbind(weights, results$solution[N+1]) } }
cc= rev/maxrev;
REVE=data.frame(maxrev, rev, cc)
colnames(REVE) = c('maxrev', 'revenue','Costeffi')
WriteXLS(REVE,"24-REVENUE-EFFI.xls", row.names = F, col.names = T)
```

Example 4.10 Consider the data set provided in Table 4.1. The revenue efficiency scores are listed in Table 4.11. In this table, maximum revenue along with current revenue of DMUs are reported.

Table 4.11 Optimal solutions for Example 4.10

DMU	MAX.REVENUE	CURRENT REVENUE	REVENUE EFFI.
1	2079.000	2079.000	1.000
2	2480.016	1632.000	0.658
3	3404.874	1557.000	0.457
4	2839.731	1725.000	0.607
5	2658.000	2658.000	1.000
6	2362.667	990.000	0.419
7	2039.739	1530.000	0.750
8	2838.000	2838.000	1.000
9	3539.182	1923.000	0.543
10	7062.198	1314.000	0.186

4.8 Malmquist Productivity Index—CCR Model with R Codes

In this section, DEA models to determine the progress or regress of units are formulated and then R code for each model is provided.

4.8.1 Malmquist Productivity Index—CCR Multiplier Model with R Code

Foe deriving the progress or regress of DMUs in two time periods (t and $t + 1$), Caves et al. [7] introduced an index based on DEA efficiency scores. This index is consisting of efficiency and technology variations. Vector and matrix forms of this model are as follows:

$$\begin{aligned} D_{t,t+1} &= \max \sum_{i=1}^{m} v_i x_{ip}^{t} \\ \text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ij}^{t+1} - \sum_{r=1}^{s} u_r y_{rj}^{t+1} \le 0, \quad i = 1, \ldots, n,\ j \ne p, \\ & \sum_{i=1}^{m} v_i x_{ip}^{t} - \sum_{r=1}^{s} u_r y_{rp}^{t} \ge 0, \qquad j = p, \\ & \sum_{r=1}^{s} u_r y_{rp}^{t} = 1, \\ & u_r \ge 0,\ v_i \ge 0, \qquad r = 1, \ldots, s,\ i = 1, \ldots, m. \end{aligned} \tag{4.27}$$

$$\begin{aligned} D_{t,t} &= \min \sum_{i=1}^{m} v_i x_{ip}^{t} \\ \text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ij}^{t} - \sum_{r=1}^{s} u_r y_{rj}^{t} \ge 0, \quad j = 1, \ldots, n, \\ & \sum_{r=1}^{s} u_r y_{rp}^{t} = 1, \\ & u_r \ge 0,\ v_i \ge 0, \qquad r = 1, \ldots, s,\ i = 1, \ldots, m. \end{aligned} \tag{4.28}$$

$$
\begin{aligned}
&D_{t+1,t+1} = \min \sum_{i=1}^{m} v_i x_{ip}^{t+1} \\
&\text{s.t.} \quad \sum_{i=1}^{m} v_i x_{ij}^{t+1} - \sum_{r=1}^{s} u_r y_{rj}^{t+1} \geq 0, \quad j = 1, \ldots, n, \\
&\qquad \sum_{r=1}^{s} u_r y_{rp}^{t+1} = 1, \\
&\qquad v_r \geq 0,\ u_i \geq 0, \qquad\qquad i = 1, \ldots, m,\ r = 1, \ldots, s.
\end{aligned} \tag{4.29}
$$

$$
\begin{aligned}
&D_{t+1,t} = \min \sum_{i=1}^{m} v_i x_{ip}^{t+1} \\
&\text{s.t.} \quad \sum_{i=1}^{m} v_i x_{ij}^{t} - \sum_{r=1}^{s} u_r y_{rj}^{t} \geq 0, \qquad j = 1, \ldots, n,\ j \neq p, \\
&\qquad \sum_{i=1}^{m} v_i x_{ij}^{t+1} - \sum_{r=1}^{s} u_r y_{rj}^{t+1} \geq 0, \quad j = p, \\
&\qquad \sum_{r=1}^{s} u_r y_{rj}^{t+1} = 1, \\
&\qquad v_r \geq 0,\ u_i \geq 0, \qquad\qquad i = 1, \ldots, m,\ r = 1, \ldots, s.
\end{aligned} \tag{4.30}
$$

$$
\begin{aligned}
&\mathrm{D}_{\mathrm{t,t+1}} = \min \left[x_{1p}^{t}, \ldots, x_{mp}^{t}, 0, \ldots, 0 \right] \left[v_1, \ldots, v_m, u_1, \ldots, u_s \right]^{\mathrm{t}} \\
&\text{s.t.} \quad \left[\begin{array}{ccc:ccc}
x_{11}^{t} & \cdots & x_{1n}^{t} & -y_{11}^{t} & \cdots & -y_{1n}^{t} \\
x_{p1}^{t+1} & \cdots & x_{pn}^{t+1} & -y_{p1}^{t+1} & \cdots & -y_{pn}^{t+1} \\
x_{mn}^{t} & \cdots & x_{mn}^{t} & -y_{1n}^{t} & \cdots & -y_{sn}^{t} \\
\hdashline
0 & \cdots & 0 & y_{1p}^{t+1} & \cdots & y_{sp}^{t+1}
\end{array}\right]
\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \end{array}\right]
\begin{array}{l} \geq_n \\ =_1 \end{array}
\left[\begin{array}{c} 0_{s\times 1} \\ \hdashline 1 \end{array}\right] \\
&\qquad u_r \geq 0,\ v_i \geq 0, \quad r = 1, \ldots, s,\ i = 1, \ldots, m.
\end{aligned} \tag{4.31}
$$

$$
\begin{aligned}
&\mathrm{D}_{\mathrm{t,t}} = \min \left[x_{1p}^{t}, \ldots, x_{mp}^{t}, 0, \ldots, 0 \right] \left[v_1, \ldots, v_m, u_1, \ldots, u_s \right]^{\mathrm{t}} \\
&\text{s.t.} \quad \left[\begin{array}{ccc:ccc}
x_{11}^{t} & \cdots & x_{1n}^{t} & -y_{11}^{t} & \cdots & -y_{1n}^{t} \\
\vdots & & \vdots & \vdots & & \vdots \\
x_{mn}^{t} & \cdots & x_{mn}^{t} & -y_{sn}^{t} & \cdots & -y_{sn}^{t} \\
\hdashline
0 & \cdots & 0 & y_{1p}^{t} & \cdots & y_{sp}^{t}
\end{array}\right]
\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \end{array}\right]
\begin{array}{l} \geq_n \\ =_1 \end{array}
\left[\begin{array}{c} 0_{n\times 1} \\ \hdashline 1 \end{array}\right] \\
&\qquad u_r \geq 0,\ v_i \geq 0, \quad r = 1, \ldots, s,\ i = 1, \ldots, m.
\end{aligned} \tag{4.32}
$$

$$\mathrm{D}_{\mathrm{t+1,t+1}} = \min\left[x_{1p}^{t+1},...,x_{mp}^{t+1},0,...,0\right]\left[v_1,...,v_m,u_1,...,u_s\right]^{\mathrm{t}}$$

$$\text{s.t.} \quad \left[\begin{array}{ccc:ccc} x_{11}^{t+1} & \cdots & x_{1n}^{t+1} & -y_{11}^{t+1} & \cdots & -y_{1n}^{t+1} \\ \vdots & & \vdots & \vdots & & \vdots \\ x_{m1}^{t+1} & \cdots & x_{mn}^{t+1} & -y_{sn}^{t+1} & \cdots & -y_{sn}^{t+1} \\ \hdashline 0 & \cdots & 0 & y_{1p}^{t+1} & \cdots & y_{sp}^{t+1} \end{array}\right]\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ u_1 \\ \vdots \\ u_s \end{array}\right] \begin{array}{l} \geq_n \\ =_1 \end{array} \left[\begin{array}{c} 0_{n\times 1} \\ \hdashline 1 \end{array}\right] \tag{4.33}$$

$$u_r \geq 0,\ v_i \geq 0, \quad r = 1,...,s,\ i = 1,...,m.$$

$$\mathrm{D}_{\mathrm{t+1,t}} = \min\left[x_{1p}^{t+1},...,x_{mp}^{t+1},0,...,0\right]\left[v_1,...,v_m,u_1,...,u_s\right]^{\mathrm{t}}$$

$$\text{s.t.} \quad \left[\begin{array}{ccc:ccc} x_{11}^{t} & \cdots & x_{1n}^{t} & -y_{11}^{t} & \cdots & -y_{1n}^{t} \\ \vdots & & & \vdots & & \\ x_{p1}^{t+1} & \cdots & x_{pn}^{t+1} & -y_{p1}^{t+1} & \cdots & -y_{pn}^{t+1} \\ \vdots & & & \vdots & & \\ x_{m1}^{t} & \cdots & x_{mn}^{t} & -y_{s1}^{t} & \cdots & -y_{sn}^{t} \\ \hdashline 0 & \cdots & 0 & y_{1p}^{t+1} & \cdots & y_{sp}^{t+1} \end{array}\right]\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ u_1 \\ \vdots \\ u_s \end{array}\right] \begin{array}{l} \geq_n \\ =_1 \end{array} \left[\begin{array}{c} 0_{n\times 1} \\ \hdashline 1 \end{array}\right] \tag{4.34}$$

$$u_r \geq 0,\ v_i \geq 0, \quad r = 1,...,s,\ i = 1,...,m.$$

Now using the following index, it is possible to obtain the status of progress or regress of DMU_p under evaluation. Consider

$$M_p = \left[\frac{D_p^t(x_p^{t+1}, y_p^{t+1})}{D_p^t(x_p^t, y_p^t)} \times \frac{D_p^{t+1}(x_p^{t+1}, y_p^{t+1})}{D_p^{t+1}(x_p^t, y_p^t)}\right]^{1/2}$$

- If $M_p > 1$ then DMU_p has made a progress.
- If $M_p < 1$ then DMU_p has made a regress.
- If $M_p = 1$ then DMU_p has made no a progress or a regress.

The R code of models (4.31)–(4.34) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
df1=data.frame(read_excel(path = "data2.xlsx", sheet = "1"))
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
# defining right hand side and directions
f.rhs11 = c(rep(0,N),1)
f.rhs22 = c(rep(0,N),1)
f.rhs12 = c(rep(0,N),1)
f.rhs21 = c(rep(0,N),1)
f.dir11 = c(rep(">=",N),"=")
f.dir22 = c(rep(">=",N),"=")
f.dir12 = c(rep(">=",N),"=")
f.dir21 = c(rep(">=",N),"=")
# defining matrixes of coefficient
con1.11 = cbind(df[,1:m],-df[,(m+1):(m+s)])
con1.22 = cbind(df1[,1:m],-df1[,(m+1):(m+s)])
con1.12 = cbind(df1[,1:m],-df1[,(m+1):(m+s)])
con1.21 = cbind(df[,1:m],-df[,(m+1):(m+s)])
for (j in 1:N) {
# defining matrixes of coefficient and objectives
con1.12[j,] = cbind(df[j,1:m],-df[j,(m+1):(m+s)])
con1.21[j,] = cbind(df1[j,1:m],-df1[j,(m+1):(m+s)])
f.obj11 = c(df[j,(1):(m)], rep(0,s))
f.obj22 = c(df1[j,(1):(m)], rep(0,s))
f.obj12 = c(df[j,(1):(m)], rep(0,s))
f.obj21 = c(df1[j,(1):(m)], rep(0,s))
con2.11 = c(rep(0,m), as.numeric(df[j,(m+1):(m+s)]))
con2.22 = c(rep(0,m), as.numeric(df1[j,(m+1):(m+s)]))
con2.12 = c(rep(0,m), as.numeric(df[j,(m+1):(m+s)]))
con2.21 = c(rep(0,m), as.numeric(df1[j,(m+1):(m+s)]))
f.con11 = rbind(con1.11,con2.11)
f.con22 = rbind(con1.22,con2.22)
f.con12 = rbind(con1.12,con2.12)
```

```
f.con21 = rbind(con1.21,con2.21)
# solving models
results11 = lp ("min",f.obj11, f.con11, f.dir11, f.rhs11, scale=0, compute.
sens=TRUE)
results22 = lp ("min",f.obj22, f.con22, f.dir22, f.rhs22, scale=0, compute.
sens=TRUE)
results12 = lp ("min",f.obj12, f.con12, f.dir12, f.rhs12, scale=0, compute.
sens=TRUE)
results21 = lp ("min",f.obj21, f.con21, f.dir21, f.rhs21, scale=0, compute.
sens=TRUE)
if (j==1) {
eff11 = results11$objval
eff22 = results22$objval
eff12 = results12$objval
eff21 = results21$objval
MALM = sqrt((results21$objval*results22$objval)/(results12$objval*result-
s11$objval))
} else {
eff11 = rbind(eff11, results11$objval)
eff22 = rbind(eff22, results22$objval)
eff12 = rbind(eff12, results12$objval)
eff21 = rbind(eff21, results21$objval)
MALM = rbind(MALM, sqrt((results21$objval*results22$objval)/(results12
$objval*results11$objval))) } }
MPR = data.frame(eff11,eff22, eff12, eff21, MALM)
colnames(MPR) = c('T11', "T22", "T12", "T21", "Malmquist")
WriteXLS(MPR,"25-malm-multi.xls", row.names = F, col.names = T)
```

Example 4.11 Consider the data sets provided in Tables 4.1 and 4.12. Each dataset contains ten DMUs with two inputs and two outputs in two different time intervals (time t and time $t + 1$). The Malmquist indexes are listed in Table 4.13. In this table, optimal objective function of models (4.31)–(4.34) are reported.

4.8.2 Malmquist Productivity Index—CCR Envelopment Model with R Code

Caves et al. [7] in order to distinguish the status of progress or regress of DMUs in two times periods (t and $t + 1$) used the CCR envelopment model. Consider the vector and matrix forms of this model as follows:

Table 4.12 Data2 (input and output data)

DMU	I_1	I_2	O_1	O_2
1	34	23	23	156
2	24	54	93	34
3	54	76	78	65
4	63	54	123	87
5	45	17	76	92
6	76	71	65	87
7	12	83	98	127
8	56	67	348	368
9	34	93	345	98
10	87	25	135	92

Table 4.13 Optimal values and Malmquist index

DMU	T_{11}	T_{22}	T_{12}	T_{21}	MALM. INDEX
1	1.000	1.000	1.345	1.000	0.862
2	1.313	2.299	2.963	1.000	0.769
3	1.040	4.675	2.804	1.694	1.648
4	1.600	2.292	2.379	1.000	0.776
5	1.000	1.045	1.769	1.000	0.769
6	2.274	4.705	4.264	2.281	1.052
7	1.120	1.000	1.415	1.000	0.794
8	1.000	1.000	1.423	1.000	0.838
9	1.227	1.000	1.227	1.000	0.815
10	4.960	1.000	4.960	1.000	0.202

$$
\begin{aligned}
&D_{\mathrm{t,t+1}} = \max \varphi \\
&\text{s.t.} \quad \sum_{j=1}^{n} \lambda_j x_{ij}^{t+1} \le x_{ip}^{t}, && i = 1, \ldots, m, \\
&\qquad \varphi y_{rp}^{t} - \sum_{i=1}^{u} \lambda_j y_{rj}^{t+1} \le 0, && r = 1, \ldots, s, \\
&\qquad \lambda_j \ge 0, && j = 1, \ldots, n.
\end{aligned}
\tag{4.35}
$$

$$
\begin{aligned}
&D_{t,t} = \max \varphi \\
&\text{s.t.} \quad \sum_{j=1}^{n} \lambda_j x_{ij}^{t} \le x_{ip}^{t}, && i = 1, \ldots, m, \\
&\qquad \varphi y_{rp}^{t} - \sum_{j=1}^{u} \lambda_j y_{rj}^{t} \le 0, && r = 1, \ldots, s, \\
&\qquad \lambda_j \ge 0, && j = 1, \ldots, n.
\end{aligned}
\tag{4.36}
$$

$$
\begin{aligned}
&D_{t+1,t+1} = \max \varphi \\
&\text{s.t.} \quad \sum_{j=1}^{n} \lambda_j x_{ij}^{t+1} \le x_{ip}^{t+1}, \qquad i = 1, \ldots, m, \\
&\qquad \varphi y_{rp}^{t+1} - \sum_{j=1}^{n} \lambda_j y_{rj}^{t+1} \le 0, \quad r = 1, \ldots, s, \\
&\qquad \lambda_j \ge 0, \qquad j = 1, \ldots, n.
\end{aligned} \tag{4.37}
$$

$$
\begin{aligned}
&D_{t+1,t} = \max \varphi \\
&\text{s.t.} \quad \sum_{j=1}^{n} \lambda_j x_{ij}^{t} \le x_{ip}^{t+1}, \qquad i = 1, \ldots, m, \\
&\qquad \varphi y_{rp}^{t+1} - \sum_{j=1}^{n} \lambda_j y_{rj}^{t} \le 0, \quad r = 1, \ldots, s, \\
&\qquad \lambda_j \ge 0, \qquad j = 1, \ldots, n.
\end{aligned} \tag{4.38}
$$

$$
\begin{aligned}
&\mathrm{D}_{\mathrm{t,t+1}} = \max\left[1, 0, \ldots, 0\right]\left[\varphi, \lambda_1, \ldots, \lambda_n\right]^{\mathrm{t}} \\
&\text{s.t.} \quad \left[\begin{array}{c|ccc} 0 & x_{11}^{t} & \cdots & x_{1n}^{t} \\ \vdots & \vdots & & \vdots \\ 0 & x_{m1}^{t} & \cdots & x_{mn}^{t} \\ \hline -y_{1p}^{t+1} & y_{11}^{t} & \cdots & y_{1n}^{t} \\ \vdots & \vdots & & \vdots \\ -y_{sp}^{t+1} & y_{s1}^{t+1} & \cdots & y_{sn}^{t+1} \end{array}\right] \left[\begin{array}{c} \varphi \\ \hline \lambda_1 \\ \vdots \\ \lambda_n \end{array}\right] \begin{array}{c} \le_m \\ \hline \le_s \end{array} \left[\begin{array}{c} x_{1p}^{t+1} \\ \vdots \\ x_{mp}^{t+1} \\ \hline 0_{s\times 1} \end{array}\right] \\
&\qquad \lambda_j \ge 0, \quad j = 1, \ldots, n.
\end{aligned} \tag{4.39}
$$

$$
\begin{aligned}
&\mathrm{D}_{\mathrm{t,t}} = \max\left[1, 0, \ldots, 0\right]\left[\varphi, \lambda_1, \ldots, \lambda_n\right]^{\mathrm{t}} \\
&\text{s.t.} \quad \left[\begin{array}{c|ccc} 0 & x_{11}^{t} & \cdots & x_{1n}^{t} \\ \vdots & \vdots & & \vdots \\ 0 & x_{m1}^{t} & \cdots & x_{mn}^{t} \\ \hline -y_{1p}^{t} & y_{11}^{t} & \cdots & y_{1n}^{t} \\ \vdots & \vdots & & \vdots \\ -y_{sp}^{t} & y_{s1}^{t} & \cdots & y_{sn}^{t} \end{array}\right] \left[\begin{array}{c} \varphi \\ \hline \lambda_1 \\ \vdots \\ \lambda_n \end{array}\right] \begin{array}{c} \le_m \\ \hline \le_s \end{array} \left[\begin{array}{c} x_{1p}^{t} \\ \vdots \\ x_{mp}^{t} \\ \hline 0_{s\times 1} \end{array}\right] \\
&\qquad \lambda_j \ge 0, \quad j = 1, \ldots, n.
\end{aligned} \tag{4.40}
$$

$$
\begin{aligned}
&D_{t+1,t+1} = \max\left[1,0,...,0\right]\left[\varphi,\lambda_1,...,\lambda_n\right]^{t}\\
&\text{s.t.}\quad
\left[\begin{array}{c|ccccc}
0 & x_{11}^{t+1} & \cdots & x_{1p}^{t+1} & \cdots & x_{1n}^{t+1}\\
\vdots & \vdots & & \vdots & & \vdots\\
0 & x_{m1}^{t+1} & \cdots & x_{mp}^{t+1} & \cdots & x_{mn}^{t+1}\\
\hline
-y_{1p}^{t+1} & y_{11}^{t+1} & \cdots & y_{1p}^{t+1} & \cdots & y_{1n}^{t+1}\\
\vdots & \vdots & & \vdots & & \vdots\\
-y_{sp}^{t+1} & y_{s1}^{t+1} & \cdots & y_{sp}^{t+1} & \cdots & y_{sn}^{t+1}
\end{array}\right]
\left[\begin{array}{c}\varphi\\ \hline \lambda_1\\ \vdots\\ \lambda_n\end{array}\right]
\begin{array}{c}\leq_m\\ \geq_s\end{array}
\left[\begin{array}{c}x_{1p}^{t+1}\\ \vdots\\ x_{mp}^{t+1}\\ \hline 0_{s\times 1}\end{array}\right]\\
&\lambda_j \geq 0,\quad j=1,...,n.
\end{aligned}
\tag{4.41}
$$

$$
\begin{aligned}
&D_{t+1,t} = \max\left[1,0,...,0\right]\left[\varphi,\lambda_1,...,\lambda_n\right]^{t}\\
&\text{s.t.}\quad
\left[\begin{array}{c|ccccc}
0 & x_{11}^{t+1} & \cdots & x_{1p}^{t} & \cdots & x_{1n}^{t+1}\\
\vdots & \vdots & & \vdots & & \vdots\\
0 & x_{m1}^{t+1} & \cdots & x_{mp}^{t} & \cdots & x_{mn}^{t+1}\\
\hline
-y_{1p}^{t} & y_{11}^{t+1} & \cdots & y_{1p}^{t} & \cdots & y_{1n}^{t+1}\\
\vdots & \vdots & & \vdots & & \vdots\\
-y_{sp}^{t} & y_{s1}^{t+1} & \cdots & y_{sp}^{+} & \cdots & y_{sn}^{t+1}
\end{array}\right]
\left[\begin{array}{c}\varphi\\ \hline \lambda_1\\ \vdots\\ \lambda_n\end{array}\right]
\begin{array}{c}\leq_m\\ \geq_s\end{array}
\left[\begin{array}{c}x_{1p}^{t}\\ \vdots\\ x_{mp}^{t}\\ \hline 0_{s\times 1}\end{array}\right]\\
&\lambda_j \geq 0, j=1,...,n.
\end{aligned}
\tag{4.42}
$$

The R code of models (4.39)–(4.42) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
df1=data.frame(read_excel(path = "data2.xlsx", sheet = "1"))
# defining matrixes to be filled wit data
f.con=matrix(ncol=N+1,nrow=m+s)
f.con1=matrix(ncol=N+1,nrow=m+s)
f.con2=matrix(ncol=N+1,nrow=m+s)
f.con3=matrix(ncol=N+1,nrow=m+s)
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
```

```
for (j in 1:N) {
# defining right hand sides, directions, and objectives
f.rhs = c(unlist(unname(df[j,(1):(m)])), rep(0,s))
f.rhs1 = c(unlist(unname(df1[j,(1):(m)])), rep(0,s))
f.rhs2 = c(unlist(unname(df[j,(1):(m)])), rep(0,s))
f.rhs3= c(unlist(unname(df1[j,(1):(m)])), rep(0,s))
f.dir = c(rep("<=",m),rep("<=",s))
f.dir1 = c(rep("<=",m),rep("<=",s))
f.dir2 = c(rep("<=",m),rep("<=",s))
f.dir3 = c(rep("<=",m),rep("<=",s))
f.obj = c(1, rep(0,N))
f.obj1 = c(1, rep(0,N))
f.obj2 = c(1, rep(0,N))
f.obj3 = c(1, rep(0,N))
for(i in 1:m){
f.con[i,1:(N+1)] =c(0,df[,i])
f.con1[i,1:(N+1)]=c(0,df1[,i])
f.con2[i,1:(N+1)]=c(0,df1[,i])
f.con3[i,1:(N+1)]=c(0,df[,i]) }
for(r in (m+1):(s+m)){
f.con[r,1:(N+1)] =c(as.numeric(df[j,r]),as.numeric(-1*df[,r]))
f.con1[r,1:(N+1)]=c(as.numeric(df1[j,r]),as.numeric(-1*df1[,r]))
f.con2[r,1:(N+1)]=c(as.numeric(df[j,r]),as.numeric(-1*df1[,r]))
f.con3[r,1:(N+1)]=c(as.numeric(df1[j,r]),as.numeric(-1*df[,r])) }
# solving models
results = lp('max', as.numeric(f.obj), f.con, f.dir, f.rhs, scale=0, compute.
sens=TRUE)
results1 = lp('max', as.numeric(f.obj1), f.con1, f.dir1, f.rhs1, scale=0,
compute.sens=TRUE)
results2 = lp('max', as.numeric(f.obj2), f.con2, f.dir2, f.rhs2, scale=0,
compute.sens=TRUE)
results3 = lp('max', as.numeric(f.obj3), f.con3, f.dir3, f.rhs3, scale=0,
compute.sens=TRUE)
if (j==1) { MALM = sqrt((results1$objval*results3$objval)/(results$obj-
val*results2$objval))
} else { MALM = rbind(MALM,(sqrt((results1$objval*results3$objval)/(re-
sults$objval*results2$objval)))) } }
TFP=data.frame(MALM)
colnames(TFP) = c('MALMQUIST')
WriteXLS(TFP,"26-MALM-ENV-CCR-OUT.xls", row.names = F,
col.names = T)
```

Table 4.14 Malmquist index

DMU	MALM. INDEX
1	0.563
2	0.708
3	1.666
4	0.746
5	0.532
6	1.030
7	0.470
8	0.436
9	0.317
10	0.108

Example 4.12 Consider the data sets provided in Tables 4.1 and 4.12. The Malmquist indexes are listed in Table 4.14.

4.9 SBM Models with R Codes

In this section, a slacks-based measure (SBM) of efficiency in DEA is presented [8]. This scalar measure deals directly with the input excesses and the output shortfalls of the DMU concerned.

4.9.1 First Model of SBM with R Code

In the case that the numerator of the fractional objective function in SBM model is transformed to the constraints, the following model is obtained:

$$
\begin{aligned}
\max \quad & 1+\tfrac{1}{s}\sum_{r=1}^{s}\tfrac{s_r^+}{y_{rp}} \\
\text{s.t.} \quad & -tx_{ip}+\sum_{j=1}^{u}\lambda_j x_{ij}+s_i^-=0, \quad i=1,\ldots,m, \\
& -ty_{rp}+\sum\lambda_j y_{rj}-s_r^+=0, \quad r=1,\ldots,s, \\
& t-\tfrac{1}{m}\sum_{i=1}^{m}\tfrac{s_i^-}{x_{ip}}=1, \\
& s_i^-\geq 0,\ s_r^+\geq 0, \qquad i=1,\ldots,m,\ r=1,\ldots,s, \\
& t\geq\varepsilon,\ \lambda_j\geq 0, \qquad j=1,\ldots,n.
\end{aligned}
\tag{4.43}
$$

The matrix form of model (4.43) is as follows:

$$\max \quad \left[1,0,...,0,0,...,0,\frac{1}{sy_{1p}},...,\frac{1}{sy_{sp}}\right]\left[t,\lambda_1,...,\lambda_n,s_1^-,...,s_m^-,s_1^+,...,s_s^+\right]^{\mathrm{t}}$$

s.t.

$$\left[\begin{array}{cccc|ccc|c} -x_{1p} & x_{11} & \cdots & x_{1n} & & & & \\ \vdots & \vdots & & \vdots & & I_m & & o_{n\times m} \\ -x_{mp} & x_{m1} & \cdots & x_{mn} & & & & \\ \hline -y_{1p} & y_{11} & \cdots & y_{1n} & & & & \\ \vdots & \vdots & & \vdots & & o_{s\times m} & & -I_s \\ -y_{sp} & y_{s1} & \cdots & y_{sn} & & & & \\ \hline 1 & 0 & \cdots & 0 & \frac{-1}{mx_{ip}} & \cdots & \frac{-1}{mx_{mp}} & 0 \\ & & & & & & & \\ 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \end{array}\right]\left[\begin{array}{c} t \\ \hline \lambda_1 \\ \vdots \\ \lambda_n \\ \hline s_1 \\ \vdots \\ s_m \\ \hline s_1^+ \\ \vdots \\ s_s^+ \end{array}\right] \underset{\geq_1}{=_{m+s+1}} \left[\begin{array}{c} 0_{m\times 1} \\ \hline 0_{s\times 1} \\ \hline 1 \end{array}\right] \tag{4.44}$$

$\lambda_i \geq 0, \quad j = 1,...,n,$

$s_i^- \geq 0,\ s_r^+ \geq 0, \quad i = 1,\ldots,m,\ \mathrm{r} = 1,\ldots,s.$

The R code of model (4.44) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
# defining a matrix to be filled with data
f.con=matrix(ncol=N,nrow=m+s)
for (j in 1:N) {
# defining right hand side, directions, and objective function
f.rhs = c(rep(0,m), rep(0,s),1, 0.00000001)
f.dir = c(rep("=",m), rep("=", s), "=", ">=")
f.obj = c(1,rep(0,N), rep(0,m), 1/(as.numeric(s*df[j,(m+1):(m+s)])))
# defining matrix of constraints' coefficient
for(i in 1:m) {
```

```
f.con[i,1:N]=c(df[,i])
}
for(r in (m+1):(s+m))
{
f.con[r,1:N]=c(df[,r])
}
c1=rbind(diag(m), matrix(0,s,m))
c2=rbind(matrix(0,m,s),-diag(s))
c3 = cbind(f.con, c1, c2)
c4= c(as.numeric(t(-df[j,1:m])), as.numeric(t(-df[j,(m+1):(m+s)])))
con5=cbind(c4, c3)
con6= c(1,rep(0,N),(-1)/(as.numeric(m*df[j,1:m])),rep(0,s))
con7= c(1,rep(0,N),0,rep(0,s))
final.con = rbind(con5, con6,con7)
# solving model
resultsbm = lp('max', f.obj, final.con, f.dir, f.rhs, scale=0, compute.
sens=TRUE)
if (j==1)
sbmeff = resultsbm$objval
else
sbmeff = rbind(sbmeff,resultsbm$objval)
}
sbm=data.frame(sbmeff)
colnames(sbm) = c('SBM');sbm
WriteXLS(sbm,"SBM", row.names = FALSE, col.names = FALSE)
```

Example 4.13 Consider the data set provided in Table 4.1. The efficiency scores obtained from linearization of SBM model are listed in Table 4.15.

Table 4.15 Efficiency

DMU	EFFI.
1	1.680
2	3.355
3	1.975
4	1.000
5	3.239
6	1.979
7	1.000
8	2.319
9	6.627
10	1.680

4.9.2 Second Model of SBM with R Code

In the case that the denominator of the fractional objective function in SBM model proposed by Tone [8] is transformed to the constraints, the following model is obtained:

$$\begin{aligned}
\min \quad & 1-\frac{1}{m}\sum_{i=1}^{m}\frac{s_r^-}{x_{ip}} \\
\text{s.t.} \quad & -tx_{ip}+\sum_{j=1}^{u}\lambda_j x_{ij}+s_i^- = 0, \quad i=1,\ldots,m, \\
& -ty_{rp}+\sum_{j=1}^{u}\lambda_j y_{rj}-s_r^+ = 0, \quad r=1,\ldots,s, \\
& t+\frac{1}{m}\sum_{r=1}^{s}\frac{s_i^+}{y_{rp}}=1, \\
& \lambda_j \geq 0, \qquad\qquad j=1,\ldots,n, \\
& s_i^- \geq 0,\ s_r^+ \geq 0, \qquad i=1,\ldots,m,\ r=1,\ldots,s, \\
& t \geq \varepsilon.
\end{aligned} \tag{4.45}$$

The matrix form of model (4.45) can be stated as follows:

$$\min \quad \left[1,0,\ldots,0,\frac{-1}{mx_{1p}},\ldots,\frac{-1}{mx_{mp}},0,\ldots,0\right]\left[t,\lambda_1,\ldots,\lambda_n,s_1^-,\ldots,s_m^-,s_1^+,\ldots,s_s^+\right]^{\mathrm{t}}$$

$$\text{s.t.} \quad \left[\begin{array}{c|ccc|c|cccc}
-x_{1p} & x_{11} & \cdots & x_{1n} & I_m & o_{m\times s} \\
\vdots & \vdots & & \vdots & & \\
-x_{mp} & x_{m1} & \cdots & x_{mn} & & \\
\hline
-y_{1p} & y_{11} & \cdots & y_{1n} & o_{s\times m} & -I_s \\
\vdots & \vdots & & \vdots & & \\
-y_{sp} & y_{s1} & \cdots & y_{sn} & & \\
\hline
1 & 0 & \cdots & 0 & o_{1\times m} & \frac{1}{sy_{1p}} & \cdots & \frac{1}{s_{ys}} \\
1 & 0 & \ldots & 0 & o_{1\times m} & 0 & \ldots & 0
\end{array}\right]\left[\begin{array}{c} t \\ \hline \lambda_1 \\ \vdots \\ \lambda_n \\ \hline s_1^- \\ \vdots \\ s_m^- \\ \hline s_1^+ \\ \vdots \\ s_s^+ \end{array}\right] =_{m+s+1} \left[\begin{array}{c} 0_{m\times 1} \\ \hline 0_{s\times 1} \\ \hline 1 \\ \varepsilon \end{array}\right] \tag{4.46}$$

$$\geq_1$$

$$\lambda_j \geq 0, \quad j=1,\ldots,n,$$

$$s_i^- \geq 0,\ s_r^+ \geq 0, \quad i=1,\ldots,m,\ r=1,\ldots,s.$$

The R code of model (4.46) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df=data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
# defining a matrix to be filled with data
f.con=matrix(ncol=N,nrow=m+s)
for (j in 1:N) {
# defining right hand side, directions, and objective
f.rhs = c(rep(0,m), rep(0,s),1,0.0000000000001)
f.dir = c(rep("=",m), rep("=", s), "=",">=")
f.obj = c(1,rep(0,N),(-1)/(as.numeric(m*df[j,1:m])),rep(0,s))
# defining matrix of constraints' coefficient
for(i in 1:m) {
f.con[i,1:N]=c(df[,i])
}
for(r in (m+1):(s+m))
{
f.con[r,1:N]=c(df[,r])
}
c1=rbind(diag(m), matrix(0,s,m))
c2=rbind(matrix(0,m,s),-diag(s))
c3 = cbind(f.con, c1, c2)
c4= c(as.numeric(t(-df[j,1:m])), as.numeric(t(-df[j,(m+1):(m+s)])))
con5=cbind(c4, c3)
con6= c(1,rep(0,N), rep(0,m), 1/(as.numeric(s*df[j,(m+1):(m+s)])))
con7= c(1,rep(0,N),0,rep(0,s))
final.con = rbind(con5, con6)
# solving model
resultsbm = lp('min', f.obj, final.con, f.dir, f.rhs, scale=0, compute.
sens=TRUE)
if (j==1)
sbmeff = resultsbm$objval
else
sbmeff = rbind(sbmeff,resultsbm$objval)
```

```
}
sbm=data.frame(sbmeff)
colnames(sbm) = c('SBM')
WriteXLS(sbm,"SBM", row.names = FALSE, col.names = FALSE)
```

Example 4.14 Consider the data set provided in Table 4.1. The efficiency scores obtained from linearization of SBM model are listed in Table 4.16.

4.10 Series Network DEA Model with R Code

Färe and Grosskopf [9, 10] introduced network DEA model. They considered intermediate products between stages one and two. Consider the vector and matrix forms of this model as follows:

$$\begin{aligned}
\max \quad & \sum_{r=1}^{s} u_i y_{rp} \\
\text{s.t.} \quad & \sum_{r=1}^{s} u_r y_{rj} - \sum_{L=1}^{l} w_l z_{Lj} \le 0, \quad j = 1, \ldots, n, \\
& \sum_{L=1}^{l} w_l z_{Lj} - \sum_{i=1}^{m} v_i x_{ij} \le 0, \quad k = 1, \ldots, n, \\
& \sum_{i=1}^{m} v_i x_{ip} = 1, \\
& v_i \ge 0, \; u_r \ge 0, \; z_L \ge 0 \qquad i = 1, \ldots, m, \; r = 1, \ldots, s, \; L = 1, \ldots, l.
\end{aligned} \tag{4.47}$$

Table 4.16 Efficiency

DMU	EFFI.
1	0.595
2	0.298
3	0.506
4	1.000
5	0.309
6	0.505
7	1.000
8	0.431
9	0.151
10	0.595

$$\max \quad \left[y_{1p},...,y_{sp},0,...,0,0,...,0\right]\left[u_1,...,u_s,v_1,...,v_m,1,...,w_l\right]^{\mathrm{t}}$$

$$\text{s.t.} \quad \left[\begin{array}{ccc:ccc:ccc} y_{11} & \cdots & y_{s1} & & & & -z_{11} & \cdots & -z_{l1} \\ \vdots & & \vdots & & o_{n\times m} & & \vdots & & \vdots \\ y_{1n} & \cdots & y_{sn} & & & & -z_{1n} & \cdots & -z_{ln} \\ \hdashline & & & -x_{11} & \cdots & -x_{m1} & z_{11} & \cdots & z_{l1} \\ & o_{n\times m} & & \vdots & & \vdots & \vdots & & \vdots \\ & & & -x_{1n} & \cdots & -x_{mn} & z_{1n} & \cdots & z_{ln} \\ \hdashline & o_{1\times s} & & x_{1p} & \cdots & x_{mp} & 0 & \cdots & 0 \end{array}\right]\left[\begin{array}{c} u_1 \\ \vdots \\ u_s \\ \hdashline v_1 \\ \vdots \\ v_m \\ \hdashline w_1 \\ \vdots \\ w_l \end{array}\right] \begin{array}{c} \le_n \\ \le_n \\ =_1 \end{array} \left[\begin{array}{c} 0_{n\times 1} \\ \hdashline 0_{n\times 1} \\ \hdashline 1 \end{array}\right] \tag{4.48}$$

$$v_i \ge 0,\ u_r \ge 0,\ z_L \ge 0, \quad i=1,...,m,\ r=1,...,s,\ L=1,...,l.$$

The R code of model (4.48) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df = data.frame(read_excel(path = "data3.xlsx", sheet="1"))
# number of elements
m=2
s=2
l = ncol(df)-m-s
N = nrow(df)
# defining inputs, outputs, intermediates to be used for deriving targets
inputs = data.matrix(df[1:2])
outputs = data.matrix(df[3:4])
intermed = data.matrix(df[5])
# defining right hand side and directions
f.rhs = c(rep(0,N), rep(0,N), 1)
f.dir = c(rep("<=",N), rep("<=",N),"=")
for (j in 1:N) {
# defining matrix of coefficient
f.obj = c(outputs[j,],rep(0,m), rep(0,l))
con1 = cbind(outputs, matrix(0,N,m), -intermed)
con2 = cbind(matrix(0,N,s), -inputs, intermed)
con3 = c(rep(0,s), inputs[j,], rep(0,l))
```

```
f.con = rbind(con1,con2, con3)
# solving model
results = lp ("max",f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=TRUE)
if (j==1) {
effcrs = results$objval
lambdas = results$duals[seq(1,N)]
weights= results$solution
} else {
effcrs = rbind(effcrs, results$objval)
lambdas = rbind(lambdas, results$duals[seq(1,N)])
weights= rbind(rweights, esults$solution) } }
ls = data.frame(effcrs, weights)
colnames(ls) = c('effi', "weights")
WriteXLS(ls,"29-network.xls", row.names = F, col.names = T)
```

Example 4.15 Consider the data set provided in Table 4.1. The efficiency scores of the network as well as optimal weights are listed in Table 4.17.

As "compute.sense=T" thus it is possible to obtain dual variables as listed in Table 4.18.

4.11 Profit Efficiency DEA Model with R Code

The vector-based and matrix-based forms of the profit efficiency DEA model are as follows.

Table 4.17 Efficiency and optimal solutions

DMU	EFFI.	u_1*	u_2*	v_1*	v_2*	w*
1	0.592	0.000	0.008	0.041	0.014	0.022
2	0.332	0.007	0.001	0.000	0.031	0.015
3	0.397	0.005	0.000	0.022	0.008	0.012
4	0.302	0.000	0.005	0.025	0.009	0.014
5	0.471	0.005	0.000	0.021	0.007	0.011
6	0.155	0.000	0.004	0.063	0.000	0.013
7	0.359	0.000	0.006	0.030	0.010	0.016
8	0.651	0.006	0.001	0.000	0.029	0.014
9	0.377	0.005	0.000	0.020	0.007	0.011
10	0.098	0.000	0.002	0.000	0.011	0.005

Table 4.18 Optimal values of Lambda

DMU	LAN_1*	LAN_2*	LAN_3*	LAN_4*	LAN_5*	LAN_6*	LAN_7*	LAN_8*	LAN_9*	LAN_{10}*
1	0.000	0.000	0.000	0.000	0.000	2.229	0.000	0.000	0.000	0.000
2	0.000	0.000	0.119	0.000	0.000	1.461	0.000	0.000	0.000	0.000
3	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000	1.857	0.000	0.000	0.000	0.000
5	0.000	0.000	0.579	0.000	0.000	1.775	0.000	0.000	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000	1.857	0.000	0.000	0.000	0.000
8	0.000	0.000	0.784	0.000	0.000	1.634	0.000	0.000	0.000	0.000
9	0.000	0.000	0.708	0.000	0.000	0.829	0.000	0.000	0.000	0.000
10	0.000	0.000	0.000	0.000	0.000	1.514	0.000	0.000	0.000	0.000

$$
\begin{aligned}
\max \quad & \sum_{r=1}^{s} w_i y_i - \sum_{i=1}^{m} c_i x_i \\
\text{s.t.} \quad & \sum_{j=1}^{n} \lambda_j x_{ij} \le x_i, && i = 1, \ldots, m, \\
& \sum_{j=1}^{n} \lambda_j y_{rj} \ge y_r, && r = 1, \ldots, s, \\
& x_i \le x_{ip}, && i = 1, \ldots, m, \\
& y_r \le y_{rp}, && r = 1, \ldots, s, \\
& \lambda_j \ge 0, && j = 1, \ldots, n, \\
& y_r \ge 0, && r = 1, \ldots, s, \\
& x_i \ge 0, && i = 1, \ldots, m.
\end{aligned}
\tag{4.49}
$$

$$
\begin{aligned}
\max \quad & \left[0,\ldots,0,-c_1,\ldots,-c_m,w_1,\ldots,w_s\right]\left[\lambda_1,\ldots,\lambda_n,x_1,\ldots,x_m,y_1,\ldots,y_s\right]^{\mathrm{t}} \\
\text{s.t.} \quad & \left[\begin{array}{ccc|c|c}
x_{11} & \cdots & x_{1n} & & \\
\vdots & & \vdots & -I_m & o_{m\times s} \\
x_{m1} & \cdots & x_{mn} & & \\
\hline
y_{11} & \cdots & y_{1n} & & \\
\vdots & & \vdots & o_{s\times m} & -I_s \\
y_{s1} & \cdots & y_{sn} & & \\
\hline
\multicolumn{3}{c|}{o_{m\times n}} & I_m & o_{m\times s} \\
\hline
\multicolumn{3}{c|}{o_{s\times n}} & o_{s\times m} & I_s
\end{array}\right]
\left[\begin{array}{c}
\lambda_1 \\ \vdots \\ \lambda_n \\ \hline x_1 \\ \vdots \\ x_m \\ \hline y_1 \\ \vdots \\ y_m
\end{array}\right]
\begin{array}{c}
\ge_s \\ \hline \le_m \\ \hline \ge_s
\end{array}
\left[\begin{array}{c}
0_{m\times 1} \\ \hline 0_{s\times 1} \\ \hline x_{1p} \\ \vdots \\ x_{mp} \\ \hline y_{1p} \\ \vdots \\ y_{sp}
\end{array}\right] \\
& \lambda_j \ge 0, \quad j = 1,\ldots,n, \\
& y_r \ge 0, \quad r = 1,\ldots,s, \\
& x_i \ge 0, \quad i = 1,\ldots,\mathrm{m}.
\end{aligned}
\tag{4.50}
$$

The R code of model (4.50) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
library(xlsxjars)
# reading data
df = data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# defining inputs and outputs
inputs = data.frame(df[1:2])
```

```
outputs=data.frame(df[3:4])
# vectors of cost and revenue
A=c(15, 21)
w=c(15,18)
for (j in 1:N) {
# defining right hand side, directions, objective, and matrix of coefficient
f.rhs = c(rep(0,m), rep(0,s), as.numeric(df[j,1:m]),as.numeric(df[j,(m+1):(m
+s)]))
f.dir = c(rep("<=",m),rep(">=",s), rep("<=",m),rep(">=",s));f.dir
f.obj = c(rep(0,N), -A, w)
c0 = rbind(t(inputs),t(outputs), matrix(0,m,N), matrix(0,s,N))
c1=rbind(-diag(m), matrix(0,s,m), diag(m), matrix(0,s,m))
c2=rbind(matrix(0,m,s), -diag(s), matrix(0,m,s), diag(s))
f.con=cbind(c0,c1, c2)
# solving model
results = lp('max', as.numeric(f.obj), f.con, f.dir, f.rhs, scale=0, compute.
sens=FALSE)
if (j==1) {
maxprofit=crossprod(w,as.matrix(t(outputs[j,])))-crossprod(A,as.matrix(t
(inputs[j,])))
op.profit = results$objval
profiteffi=maxprofit/op.profit
} else {
maxprofit=crossprod(w,as.matrix(t(outputs[j,])))-crossprod(A,as.matrix(t
(inputs[j,])))
op.profit = results$objval
D=maxprofit/op.profit
profiteffi=rbind(profiteffi,D) } }
profitefficiency=data.frame(profiteffi)
colnames(COEE) = c('profiteffi')
WriteXLS(COEE,"30-profit-EFFI.xls", row.names = F, col.names = T)
```

Example 4.16 Consider the data set provided in Table 4.1. The profit efficiency scores are listed in Table 4.19.

Table 4.19 Efficiency

DMU	PROFIT.EFFI.
1	1.000
2	0.393
3	0.081
4	0.133
5	1.000
6	−0.708
7	0.120
8	1.000
9	0.056
10	−0.387

4.12 Modified Slack Based DEA Models with R Codes

Koltai and Uzonyi-Kecskés [11] modified the SBM model proposed by Tone [8] by ignoring either the numerator or denominator of the objective function:

4.12.1 Input-Oriented Slack Based DEA Model with R Code

By considers only inputs in objective function, the vector and matrix forms of the modified SBM model are formulated as follows.

$$\begin{aligned}
\min \quad & 1 - \frac{1}{m}\sum_{i=1}^{m}\frac{s_i^-}{x_{ip}} \\
\text{s.t.} \quad & \sum_{j=1}^{n}\lambda_j x_{ij} + s_i^- = x_{ip}, \qquad i = 1,\ldots,m, \\
& \sum_{j=1}^{n}\lambda_j x_{rj} - s_r^+ = y_{rp}, \qquad r = 1,\ldots,s, \\
& \lambda_j \geq 0, \qquad j = 1,\ldots,n, \\
& s_i^- \geq 0, \qquad i = 1,\ldots,m, \\
& s_r^+ \geq 0, \qquad r = 1,\ldots,s.
\end{aligned} \tag{4.51}$$

$$\min \quad \left[0,\ldots,0,\frac{-1}{mx_{1p}},\ldots,\frac{-1}{mx_{mp}},0,\ldots,0\right]\left[\lambda_1,\ldots,\lambda_n,s_1^-,\ldots,s_m^-,s_1^+,\ldots,s_s^+\right]^{\mathrm{t}}$$

$$\text{s.t.} \quad \left[\begin{array}{ccc|c|c} x_{11} & \cdots & x_{1n} & & \\ \vdots & & \vdots & I_m & o_{m\times s} \\ x_{m1} & \cdots & x_{mn} & & \\ \hline y_{11} & \cdots & y_{1n} & & \\ \vdots & & \vdots & o_{s\times m} & -I_s \\ y_{s1} & \cdots & y_{sn} & & \end{array}\right]\left[\begin{array}{c} \lambda_1 \\ \vdots \\ \lambda_n \\ \hline s_m^- \\ \vdots \\ s_m^- \\ \hline s_1^+ \\ \vdots \\ s_s^+ \end{array}\right] =_{m+s} \left[\begin{array}{c} x_{1p} \\ \vdots \\ x_{mp} \\ \hline y_{1p} \\ \vdots \\ y_{sp} \end{array}\right] \tag{4.52}$$

$$\lambda_j \geq 0, \quad j=1,\ldots,n,$$
$$s_i^- \geq 0, \quad i=1,\ldots,m,$$
$$s_r^+ \geq 0, \quad r=1,\ldots,s.$$

The R code of model (4.52) is as follows.

```
# required libraries
library(readxl)
library(rJava)
library(xlsxjars)
library(xlsx)
library(lpSolve)
# reading data
df = data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# defining inputs and outputs
inputs = data.frame(df[1:2])
outputs=data.frame(df[3:4])
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
# definition of direction,
f.dir = c(rep("=",m), rep("=",s))
# definition of matrix of coefficient
c1=cbind(t(inputs),diag(m),matrix(0,m,s))
c2=cbind(t(outputs), matrix(0,s,m), -diag(s))
f.con = rbind(c1, c2)
```

```
for (j in 1:N){
# definition objective and right hand side
f.obj = c(rep(0,N),(-1/(m*inputs[j,])),rep(0,s))
f.rhs = c(inputs[j,],outputs[j,])
# solving model
results = lp ("min",as.numeric(f.obj), f.con, f.dir, f.rhs,scale=0)
if (j==1) {
eff.inp=(1+results$objval)
si=results$solution[(N+1):(N+m)]
so=results$solution[(N+m+1):(N+m+s)]
lambdas = results$solution[1:N]
} else {
eff.inp= rbind(eff.inp, (1+results$objval))
si=rbind(si,results$solution[(N+1):(N+m)])
so=rbind(so,results$solution[(N+m+1):(N+m+s)])
lambdas = rbind(lambdas, results$solution[1:N]) } }
russeleffi.input=data.frame(eff.inp, lambdas)
colnames(russeleffi.input) = c("effi", rep("lan", N))
WriteXLS(russeleffi.input,"31-russel-inp, row.names = F, col.names = T)
```

Example 4.17 Consider the data set provided in Table 4.1. The efficiency scores (EFFI) obtained from model (4.51) as well as optimal values of slacks (s*) are listed in Table 4.20.

Table 4.20 Efficiency and optimal values of slacks

DMU	EFFI.	S_1^-*	S_2^-*	S_1^+*	S_2^+*
1	1.000	0.000	0.000	0.000	0.000
2	0.718	9.333	5.600	0.000	22.267
3	0.932	4.221	0.000	0.000	102.056
4	0.510	7.500	34.500	0.500	0.000
5	1.000	0.000	0.000	0.000	0.000
6	0.325	8.964	64.764	0.000	0.000
7	0.647	1.500	33.500	13.500	0.000
8	1.000	0.000	0.000	0.000	0.000
9	0.809	0.667	24.200	0.000	80.533
10	0.172	61.808	72.654	6.577	0.000

4.12.2 Output-Oriented Slack Based DEA Model with R Code

By considers only outputs in objective function, the vector and matrix forms of the modified SBM model are formulated as follows.

$$\begin{aligned}
\min \quad & 1+\tfrac{1}{s}\sum_{r=1}^{s}\frac{s_r^+}{y_{rp}} \\
\text{s.t.} \quad & \sum_{j=1}^{n}\lambda_j x_{ij}+s_i^- = x_{ip}, \qquad i=1,\ldots,m, \\
& \sum_{j=1}^{n}\lambda_j y_{rj}-s_r^+ = y_{rp}, \qquad r=1,\ldots,s, \\
& \lambda_j \geq 0, \qquad j=1,\ldots,n, \\
& s_i^- \geq 0, \qquad i=1,\ldots,m, \\
& s_i^+ \geq 0, \qquad r=1,\ldots,s.
\end{aligned} \tag{4.53}$$

$$\begin{aligned}
\max \quad & \left[0,...,0,0,...,0,\frac{1}{sy_{1p}},...,\frac{1}{sy_{sp}}\right]\left[\lambda_1,...,\lambda_r,s_1^-,...,s_m^-,s_1^+,...,s_s^+\right]^t \\
\text{s.t.} \quad & \left[\begin{array}{ccc:c:c} x_{11} & \cdots & x_{1n} & & \\ \vdots & & \vdots & I_m & o_{m\times s} \\ x_{m1} & \cdots & x_{mn} & & \\ \hdashline y_{11} & \cdots & y_{1n} & & \\ \vdots & & \vdots & o_{s\times m} & -I_s \\ y_{s1} & \cdots & y_{sn} & & \end{array}\right]\left[\begin{array}{c} \lambda_1 \\ \vdots \\ \lambda_n \\ \hdashline s_1^- \\ \vdots \\ s_m^- \\ \hdashline s_1^+ \\ \vdots \\ s_s^+ \end{array}\right] =_{m+s} \left[\begin{array}{c} x_{1p} \\ \vdots \\ x_{mp} \\ \hdashline y_{1p} \\ \vdots \\ y_{sp} \end{array}\right] \\
& \lambda_j \geq 0, \quad j=1,...,n, \\
& s_i^- \geq 0, \quad i=1,...,m, \\
& s_i^+ \geq 0, \quad r=1,...,s.
\end{aligned} \tag{4.54}$$

The R code of model (4.54) is as follows.

```
# required libraries
library(readxl)
library(rJava)
library(xlsxjars)
library(xlsx)
library(lpSolve)
# definition of direction,
df = data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# defining inputs and outputs
inputs=data.frame(df[1:2])
outputs=data.frame(df[3:4])
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
# definition of direction, and matrix of coefficient
f.dir = c(rep("=",m), rep("=",s))
con1 = cbind(t(inputs),diag(m), matrix(0,m,s))
con2 = cbind(t(outputs), matrix(0,s,m),-diag(s))
f.con = rbind(con1,con2)
for (j in 1:N){
# definition of right hand side and objective
f.obj = c(rep(0,N),rep(0,m), 1/(s*outputs[j,]))
f.rhs = c(inputs[j,],outputs[j,])
# solving mode
results = lp ("max",as.numeric(f.obj), f.con, f.dir, f.rhs,scale=0)
if (j==1)
{ eff.out=1+results$objval
si=results$solution[(N+1):(N+m)]
so=results$solution[(N+m+1):(N+m+s)]
lambdas = results$solution[1:(N)]
} else {
eff.out= rbind(eff.out, 1+results$objval)
si=rbind(si,results$solution[(N+1):(N+m)])
so=rbind(so,results$solution[(N+m+1):(N+m+s)])
lambdas = rbind(lambdas, results$solution[1:N]) } }
russeleffi.out=data.frame(eff.out, lambdas)
colnames(russeleffi.out) = c("effi", rep("lan", N))
WriteXLS(russeleffi.out,"32-russel-out", row.names = F, col.names = T)
```

Example 4.18 Consider the data set provided in Table 4.1. The efficiency scores obtained from model (4.53) and optimal values of slacks are listed in Tables 4.21 and 4.22.

4.13 Congestion DEA Model with R Code

Cooper et al. [12] introduced a DEA model for distinguishing existence of congestion in inputs. The vector-based and matrix-based forms of this model are as follows:

$$\begin{aligned}
&\max \quad \varphi \\
&\text{s.t.} \quad \sum_{j=1}^{n} \lambda_j x_{ij} \le x_{io}, \qquad i = 1, \ldots, m, \\
&\qquad \sum_{j=1}^{n} \lambda_j y_{rj} \ge \varphi y_{ro}, \quad r = 1, \ldots, s, \\
&\qquad \sum_{j=1}^{n} \lambda_j = 1, \\
&\qquad \lambda_j \ge 0, \qquad j = 1, \ldots, n.
\end{aligned} \tag{4.55}$$

$$\begin{aligned}
&\max \quad \sum_{i=1}^{m} \delta_i \\
&\text{s.t.} \quad \sum_{j=1}^{n} \lambda_j x_{ij} + \delta_i = \hat{x}_{io}, \quad i = 1, \ldots, m, \\
&\qquad \sum_{j=1}^{n} \lambda_j y_{rj} = \hat{y}_{ro}, \qquad r = 1, \ldots, s, \\
&\qquad \sum_{j=1}^{n} \lambda_j = 1, \\
&\qquad \delta_i \le \hat{s}_i^-, \\
&\qquad \lambda_j,\ \delta_i \ge 0, \qquad j = 1, \ldots, n,\ i = 1, \ldots, m.
\end{aligned} \tag{4.56}$$

Note that in these models $\hat{x}_{io} = \sum_{j=1}^{n} \lambda_j^* x_{ij}$, $\hat{s}_i^- = x_{io} - \hat{x}_{io}$. Finally, according to $s_i^c = \hat{s}_i^- - \delta_i^*$ one can distinguish the congestion of inputs.

Table 4.21 Optimal values of intensifier variables

DMU	LAN_1*	LAN_2*	LAN_3*	LAN_4*	LAN_5*	LAN_6*	LAN_7*	LAN_8*	LAN_9*	LAN_{10}*
1	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	1.040	0.000	0.000	0.000	0.000	0.000	0.000	0.112	0.000	0.000
3	1.575	0.000	0.000	0.000	0.000	0.000	0.000	0.042	0.000	0.000
4	0.835	0.000	0.000	0.000	0.415	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
6	0.000	0.000	0.000	0.000	0.889	0.000	0.000	0.000	0.000	0.000
7	0.402	0.000	0.000	0.000	0.443	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
9	1.121	0.000	0.000	0.000	0.455	0.000	0.000	0.000	0.000	0.000
10	2.870	0.000	0.000	0.000	0.000	0.000	0.000	0.386	0.000	0.000

Table 4.22 Efficiency scores and optimal slack values

DMU	EFFI.	S_1^-*	S_2^-*	S_1^+*	S_2^+*
1	1.000	0.000	0.000	0.000	0.000
2	1.486	0.000	0.000	13.776	35.632
3	3.126	4.221	0.000	0.000	102.056
4	1.734	0.000	0.000	36.281	31.696
5	1.000	0.000	0.000	0.000	0.000
6	2.558	0.000	8.222	52.444	32.556
7	1.670	0.000	8.375	32.182	0.000
8	1.000	0.000	0.000	0.000	0.000
9	1.939	0.000	0.000	16.545	76.000
10	5.867	0.000	0.000	142.978	200.196

$$
\begin{aligned}
&\max \quad [1,0,\ldots,0][\varphi,\lambda_1,\ldots,\lambda_n]^{\mathrm{t}} \\
&\text{s.t.} \quad \left[\begin{array}{c|ccc} 0 & x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & & \vdots \\ 0 & x_{m1} & \cdots & x_{mn} \\ - & - & - & - \\ -y_{1p} & y_{11} & \cdots & y_{1n} \\ & \vdots & & \vdots \\ -y_{sp} & y_{s1} & \cdots & y_{sn} \\ - & - & - & - \\ 0 & 1 & \cdots & 1 \end{array}\right] \left[\begin{array}{c} \varphi \\ \lambda_1 \\ \vdots \\ \lambda_n \end{array}\right] \begin{array}{c} \leq_m \\ \geq_s \\ = \end{array} \left[\begin{array}{c} x_{1p} \\ \vdots \\ x_{mp} \\ 0_{m\times 1} \\ 1 \end{array}\right] \\
&\lambda_j, \quad j=1,\ldots,n.
\end{aligned} \tag{4.57}
$$

$$
\begin{aligned}
&\max \quad [0,\ldots,0,1,\ldots,1][\lambda_1,\ldots,\lambda_n,\delta_1,\ldots,\delta_m]^{\mathrm{t}} \\
&\text{s.t.} \quad \left[\begin{array}{ccc|c} x_{11} & \cdots & x_{1n} & \\ \vdots & & \vdots & \mathrm{I}_{m\times m} \\ x_{m1} & \cdots & x_{mn} & \\ - & - & - & - \\ y_{11} & \cdots & y_{1n} & \\ \vdots & & \vdots & 0_{s\times m} \\ y_{s1} & \cdots & y_{sn} & \\ - & - & - & - \\ 1 & \cdots & 1 & 0_{1\times m} \\ - & - & - & - \\ 0 & \cdots & 0 & \\ \vdots & & \vdots & \mathrm{I}_{m\times m} \\ 0 & \cdots & 0 & \end{array}\right] \left[\begin{array}{c} \lambda_1 \\ \vdots \\ \lambda_n \\ \delta_1 \\ \vdots \\ \delta_m \end{array}\right] \begin{array}{c} =_{m+s+1} \\ \leq_m \end{array} \left[\begin{array}{c} \hat{x}_{1o} \\ \vdots \\ \hat{x}_{mo} \\ \hat{y}_{1o} \\ \vdots \\ \hat{y}_{so} \\ 1 \\ \hat{s}_1^- \\ \vdots \\ \hat{s}_m^- \end{array}\right] \\
&\lambda_j, \delta_i \geq 0, \quad j=1,\ldots,n,\ i=1,\ldots,m.
\end{aligned} \tag{4.58}
$$

The R code of models (4.57) and (4.58) is as follows.

```
# required libraries
library(rJava)
library(readxl)
library(xlsx)
library(lpSolve)
# reading data
df = data.frame(read_excel(path = "data1.xlsx", sheet="1"))
# defining inputs and outputs
inputs=data.frame(df[1:2])
outputs=data.frame(df[3:4])
# number of elements
n=dim(df)[1]
m=dim(inputs)[2]
s=dim(outputs)[2]
# definition of direction,
f1.obj = c(1,rep(0,n))
f2.obj = c(rep(0,n),rep(1,m))
f1.dir = c(rep("<=",m),rep(">=",s),"=")
f2.dir = c(rep("=",m+s+1),rep("<=",m))
for (j in 1:n){
# definition right hand side and matrix of coefficient
f1.rhs = c(inputs[j,],rep(0,s),1)
con1 = rbind(t(inputs),t(outputs),c(rep(1,n)))
con2 = rbind(t(inputs),t(outputs),c(rep(1,n)),matrix(0,m,n))
con3 = rbind(matrix(0,m,1),t(-outputs[j,]),0)
con4 = rbind(diag(m),matrix(0,s,m),matrix(0,1,m),diag(m))
f1.con = cbind(con3,con1)
f2.con = cbind(con2,con4)
# solving model
results_BCC = lp ("max",as.numeric(f1.obj), f1.con, f1.dir, f1.rhs,scale=0)
if (j==1) {
eff = results_BCC$solution[1]
Lam = results_BCC$solution[2:(n+1)]
lambdas = results_BCC$solution[2:(n+1)]
xh = Lam %*% as.matrix(inputs)
yh = Lam %*% as.matrix(outputs)
```

```
sh = inputs[j,]-xh
f2.rhs = c(xh,yh,1,sh)
results_cong = lp ("max",as.numeric(f2.obj), f2.con, f2.dir, f2.rhs,scale=0)
cong = sh-results_cong$solution[(n+1):(n+m)]
congestion= sum(sh-results_cong$solution[(n+1):(n+m)])
} else {
eff = rbind(eff, results_BCC$solution[1])
Lam = results_BCC$solution[2:(n+1)]
lambdas = rbind(lambdas, results_BCC$solution[2:(n+1)])
xh = Lam %*% as.matrix(inputs)
yh = Lam %*% as.matrix(outputs)
sh = inputs[j,]-xh
f2.rhs = c(xh,yh,1,sh)
results_cong = lp ("max",as.numeric(f2.obj), f2.con, f2.dir, f2.rhs,scale=0)
cong = rbind(cong,sh-results_cong$solution[(n+1):(n+m)])
congestion= rbind(congestion, sum(sh-results_cong$solution[(n+1):(n
+m)])) } }
CO = data.frame(cong, congestion)
colnames(CO) = c("cong.i1", "cong.i2", "congestion")
WriteXLS(CO,"33-congestion", row.names = T, col.names = F)
```

Example 4.19 Consider the data set provided in Table 4.1. The congestion of each input and sum of them are listed in Table 4.23.

4.14 Common Set of Weights DEA Model with R Code

Hosseinzadeh et al. [13] introduced the common set of weights DEA model for evaluation of DMUs. Vector and matrix forms of this model are as follows.

$$\begin{aligned} \min \quad & \sum_{j=1}^{m} d_j \\ \text{s.t.} \quad & \sum_{r=1}^{s} u_r y_{rj} - \sum_{i=1}^{m} v_i x_{ij} + d_j = 0, \quad j = 1, \ldots, n, \\ & u_r \geq \varepsilon, \qquad\qquad\qquad\qquad r = 1, \ldots, s, \\ & v_i \geq \varepsilon, \qquad\qquad\qquad\qquad i = 1, \ldots, m, \\ & d_j^{+} \geq 0, \qquad\qquad\qquad\qquad j = 1, \ldots, n. \end{aligned} \tag{4.59}$$

Table 4.23 Congestion

DMU	CONG.I1	CONG.I2	CONGESTION
1	0.000	0.000	0.000
2	0.000	0.000	0.000
3	0.000	0.000	0.000
4	5.000	30.000	35.000
5	0.000	0.000	0.000
6	0.000	44.239	44.239
7	0.000	0.000	0.000
8	0.000	0.000	0.000
9	0.000	0.000	0.000
10	57.000	64.000	121.000

$$\begin{aligned}
&\min \quad [0,\ldots,0,0,\ldots,0,1,\ldots,1][u_1,\ldots,u_s,v_1,\ldots,v_m,d_1,\ldots,d_n]^{\mathrm{t}} \\
&\text{s.t.} \quad \left[\begin{array}{ccc|ccc|c} y_{11} & \cdots & y_{s1} & -x_{11} & \cdots & -x_{m1} & \\ \vdots & & \vdots & \vdots & & \vdots & I_n \\ y_{1n} & \cdots & y_{sn} & -x_{1n} & \cdots & -x_{mn} & \\ \hline \multicolumn{3}{c|}{I_s} & \multicolumn{3}{c|}{o_{s\times m}} & o_{s\times m} \\ \hline \multicolumn{3}{c|}{o_{m\times s}} & \multicolumn{3}{c|}{I_m} & o_{m\times n} \end{array}\right] \left[\begin{array}{c} u_1 \\ \vdots \\ u_n \\ \hline v_1 \\ \vdots \\ v_m \end{array}\right] \begin{array}{c} =_n \\ \geq_{m+s} \end{array} \left[\begin{array}{c} 0_{n\times 1} \\ \hline \varepsilon_{m+s\times 1} \end{array}\right] \qquad (4.60) \\
&\quad d_j^+ \geq 0, \quad j = 1,\ldots,\mathrm{n}.
\end{aligned}$$

The R code of model (4.60) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df = data.frame(read_excel(path = "data1.xlsx", sheet="1"))
```

```
# defining inputs and outputs
inputs=data.matrix(df[1:2])
outputs=data.matrix(df[3:4])
# numer of elements
epsilon=0.0001
m=2
s=ncol(df)-m
N = nrow(df)
# defining objective, directions, and right hand side
f.obj = c(rep(0,m+s),rep(1,N))
f.dir = c(rep("=",N),rep(">=",m+s))
f.rhs = c(rep(0,N),rep(epsilon,m+s))
# defining matrix of coefficient
con1= cbind(outputs, -inputs, diag(N))
con2= cbind(diag(s),matrix(0,s,m), matrix(0,s,N))
con3= cbind(matrix(0,m,s), diag(m), matrix(0,m,N))
f.con = rbind(con1, con2, con3)
# solving model
results = lp ("min",as.numeric(f.obj), f.con, f.dir, f.rhs, scale=0, compute.
sens=TRUE)
optimalweights = results$solution[1:(m+s)]; optimalweights
objectivefunc = results$objval; objectivefunc
```

Example 4.20 Consider the data set provided in Table 4.1. The optimal objective value and common weights are listed in Table 4.24.

4.15 Directional Efficiency DEA Model with R Code

Chamber et al. [14] introduced directional efficiency model with direction vector $(g^x, g^y) = (-x_p, y_p)$. Vector and matrix forms of this model are as follows:

Table 4.24 Optimal values

u_1*	u_2*	v_1*	v_2*	OBJ.FUN
0.0001	0.0001	0.00064	0.0001	0.1457

$$\begin{aligned} \max \quad & \theta \\ \text{s.t.} \quad & \theta x_{ip} + \sum_{j=1}^{n} \lambda_j x_{ij} \le x_{ip}, \qquad i = 1, \ldots, m, \\ & -\theta y_{rp} + \sum_{j=1}^{n} \lambda_j x_{rj} \ge y_{rp}, \qquad r = 1, \ldots, s, \\ & \lambda_j \ge 0, \qquad j = 1, \ldots, n. \end{aligned} \tag{4.61}$$

$$\begin{aligned} \max \quad & [1, 0, ..., 0][\theta, \lambda_1, ..., \lambda_n]^{\mathrm{t}} \\ \text{s.t.} \quad & \left[\begin{array}{c|ccc} x_{1p} & x_{11} & \cdots & -x_{1n} \\ \vdots & \vdots & & \vdots \\ x_m & x_{m1} & \cdots & x_{mn} \\ \hline -y_{1p} & y_{11} & \cdots & y_{1n} \\ \vdots & \vdots & & \vdots \\ -y_{sp} & y_{s1} & \cdots & y_{sn} \end{array}\right] \left[\begin{array}{c} \theta \\ \hline \lambda_1 \\ \vdots \\ \lambda_n \end{array}\right] \begin{array}{c} \le_m \\ \hline \ge_s \end{array} \left[\begin{array}{c} x_{1p} \\ \vdots \\ x_{mp} \\ \hline y_{1p} \\ \vdots \\ y_{sp} \end{array}\right] \\ & \lambda_j \ge 0, \quad j = 1, ..., n. \end{aligned} \tag{4.62}$$

The R code of model (4.62) is as follows.

```
# required libraries
library(readxl)
library(rJava)
library(xlsxjars)
library(xlsx)
library(lpSolve)
# reading data
df = data.frame(read_excel(path = "data1.xlsx", sheet = "1"))
# defining inputs and outputs
inputs=data.frame(df[1:2])
outputs=data.frame(df[3:4])
```

```
# number of elements
m=2
s=ncol(df)-m
N = nrow(df)
# defining objective and directions
f.obj=c(1,rep(0,N))
f.dir=c(rep("<=",m),(rep(">=",s)))
for (j in 1:N){
# defining right hand side, and matrix of coefficient
f.rhs=c(inputs[j,],outputs[j,])
con1=rbind(t(inputs[j,]),t(-outputs[j,]))
con2=rbind(t(inputs),t(outputs))
f.con=cbind(con1,con2)
# solving model
results=lp ("max",f.obj, f.con, f.dir, f.rhs,scale=0)
if (j==1)
{ effi=results$solution[1]
lambdas=results$solution[2:(N+1)]
} else {
effi=rbind(effi, results$solution[1])
lambdas=rbind(lambdas, results$solution[2:(N+1)]) } }
directiuonaleffi=data.frame(effi,lambdas)
colnames(directiuonaleffi) = c("effi", rep("lan", N))
WriteXLS(directiuonaleffi,"35-Deffi", row.names = F, col.names = T)
```

Example 4.21 Consider the data set in Table 4.1. The optimal objective value and optimal values of intensifier variables are listed in Table 4.25.

Table 4.25 Efficiency scores and optimal solutions

DMU	EFFI.	LAN_1*	LAN_2*	LAN_3*	LAN_4*	LAN_5*	LAN_6*	LAN_7*	LAN_8*	LAN_9*	LAN_{10}*
1	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.135	0.899	0.000	0.000	0.000	0.000	0.000	0.000	0.097	0.000	0.000
3	0.020	1.448	0.000	0.000	0.000	0.000	0.000	0.000	0.116	0.000	0.000
4	0.231	1.026	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
6	0.389	0.501	0.000	0.000	0.000	0.125	0.000	0.000	0.000	0.000	0.000
7	0.057	0.881	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
9	0.102	1.007	0.000	0.000	0.000	0.408	0.000	0.000	0.000	0.000	0.000
10	0.664	1.131	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

4.16 Conclusion

In this chapter some advanced DEA models have been reviewed and their properties have been considered. For each model, the R code has been written and illustrated by numerical examples. The readers after patiently studying this chapter are able to write their own R codes for more complicated DEA models.

References

1. Anderson, P., Peterson, N.C.: A procedure for ranking efficient units in data envelopment analysis. Manag. Sci. **39**(10), 1261–1264 (1993)
2. Mehrabian, S., Alirezaee, M.R., Jahanshahloo, G.R.: A complete efficiency ranking of decision making units in data envelopment analysis. Comput. Optim. Appl. **14**, 261–266 (1999)
3. Jahanshahloo, G.R., Hosseinzadeh Lotfi, F., Shoja, N., Tohidi, G., Razavyan, S.: Ranking using L1-norm in data envelopment analysis. Appl. Math. Comput. **153**, 215–224 (2004)
4. Banker, R.D., Cooper, W.W., Seiford, L.M., Thrall, R.M., Zhu, J.: Returns to scale in different DEA models. Eur. J. Oper. Res. **154**, 345–362 (2004)
5. Camanho, A.S., Dyson, R.G.: Cost efficiency, production and value-added models in the analysis of bank branch performance. JORS **56**(5), 483–494 (2005)
6. Wang, C.G., Fare, R., Seavert, C.F.: Revenue capacity efficiency of pear trees and its decomposition. J. Am. Soc. Hortic. Sci. **131**(1), 32–40 (2006)
7. Caves, D.W., Christensen, L.R., Diewert, W.E.: The economic theory of index numbers and the measurement of input, output and productivity. Econometrica **50**, 1393–1414 (1982)
8. Tone, K.: A slacks-based measure of efficiency in data envelopment analysis. Eur. J. Oper. Res. **130**, 498–509 (2001)
9. Färe, R., Grosskopf, S.: Network DEA. Socio-Econ. Plan. Sci. **34**, 35–49 (2000)
10. Färe, R., Grosskopf, S.: Profit efficiency, Farrell decompositions and the Mahler inequality. Econ. Lett. **57**, 283–287 (1997)
11. Koltai, T., Uzonyi-Kecskés, J.: The comparison of data envelopment analysis (DEA) and financial analysis results in a production simulation game. Acta Polytech. Hung. **14**(4), 167–185 (2017)
12. Cooper, W.W., Thompson, R.G., Thrall, R.M.: Introduction: extensions and new developments in DEA. Ann. Oper. Res. **66**, 3–45 (1996)
13. Hosseinzadeh Lotfi, F., Jahanshahloo, G.R., Memariani, A.: A method for finding common set of weights by multiple objective programming in data envelopment analysis. S. W. J. Pure Appl. Math. **1**, 44–54 (2000)
14. Chambers, R.G., Chung, Y., Färe, R.: Benefit and distance functions. J. Econ. Theory **70**, 407–419 (1996)

Chapter 5
Fuzzy Data Envelopment Analysis Models with R Codes

Abstract The conventional DEA models such as CCR and BBC models require precise input and output data, which may not always be available in real world applications. However, in real life problems, inputs and outputs are often imprecise. To deal with imprecise data, the notion of fuzziness has been introduced in DEA and so the DEA has been extended to fuzzy DEA (FDEA). In this chapter, the main approaches for solving FDEA models are classified into five groups and the mathematical approaches of each category are described briefly. Then, R codes for each FDEA model are provided. Finally, numerical examples are provided to illustrate the main advantages of R in FDEA models.

Keywords Data envelopment analysis · Fuzzy numbers · Fuzzy ranking · Possibility measure · Fuzzy arithmetic · R code

5.1 Introduction

The traditional DEA models such as CCR and BBC models require accurate measurement of both inputs and outputs. However, crisp input and output data may not always be relevant in real world applications. The observed values of the input and output data in real world problems are sometimes contains missing data, judgment data, or predictive data or in generally imprecise or vague data. In such situations, fuzzy sets theory is an ideal approach to handle uncertain input and output data in DEA by generalizing the notion of membership in a set and this leads to the concept of fuzzy DEA (FDEA) models.

In recent years, many researchers have formulated FDEA models for handling fuzzy input and output data. In this chapter, first different fuzzy DEA approaches are classified into several categories and then the main methods of each category for

F. Hosseinzadeh Lotfi et al., *Data Envelopment Analysis with R*, Studies in Fuzziness and Soft Computing 386, https://doi.org/10.1007/978-3-030-24277-0_5

evaluating the fuzzy efficiency of DMUs with the given fuzzy input and output data are explored in details. Moreover, R codes of these models alongside numerical examples are given.

The classical DEA models with fuzzy input and output data could be classified into the following five general groups [1]:

- The α-level approach: The main idea behind of this approach is to convert the FDEA model into a pair of parametric programming problems using α cuts. This technique gives the lower and upper bounds of interval efficiency for a DMU under consideration at a given α level that can be used to construct the corresponding fuzzy efficiency.
- The fuzzy ranking approach: The main ideal behind of this approach is to define fuzzy equality and inequality constraints in the FDEA model by ranking methods leading to a bi-level programming problem.
- The possibility approach: The main idea behind of this approach is to consider the fuzzy coefficients as fuzzy variables and fuzzy constraints as fuzzy events. Based on this technique, the possibilities of fuzzy constraints are determined using possibility theory.
- The fuzzy arithmetic approach: The main idea behind of this approach is to convert the fractional form of each FDEA model into several crisp fractional DEA model from the perspective of fuzzy arithmetic.
- The multi objective linear programming (MOLP) approach: The main idea behind of this approach is to convert the FDEA model into a MOLP problem and to use the lexicographic technique for solving the resulting model.

Now the classical DEA models are formulated in fuzzy environment. Suppose there are n DMUs to be evaluated, each with m fuzzy inputs and s fuzzy outputs. Let $\tilde{x}_{ij}$ $(i = 1, 2, \ldots, m)$ and $\tilde{y}_{rj}$ $(r = 1, 2, \ldots, s)$ be the fuzzy input and fuzzy output data of DMU_j $(j = 1, 2, \ldots, n)$. The fuzzy input-oriented CCR envelopment model to measure the performance of DMU_p is formulated as follows:

$$\begin{aligned} \min \quad & \theta_p \\ \text{s.t.} \quad & \sum_{j=1}^{n} \lambda_j \tilde{x}_{ij} \preceq \theta_p \tilde{x}_{ip}, \quad i = 1, 2, \ldots, m, \\ & \sum_{j=1}^{n} \lambda_j \tilde{y}_{rj} \succeq \tilde{y}_{rp}, \quad r = 1, 2, \ldots, s, \\ & \lambda_j \geq 0, \quad j = 1, 2, \ldots, n. \end{aligned} \tag{5.1}$$

The fuzzy input-oriented CCR multiplier model to measure the performance of DMU_p can be formulated as follow

$$\begin{aligned}
\max \quad & \sum_{r=1}^{s} u_r \tilde{y}_{rp} \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i \tilde{x}_{ip} = 1, \\
& \sum_{r=1}^{s} u_r \tilde{y}_{rj} - \sum_{i=1}^{m} v_i \tilde{x}_{ij} \leq 0, \quad j = 1, 2, \ldots, n, \\
& v_i \geq 0, \qquad\qquad i = 1, 2, \ldots, m, \\
& u_r \geq 0, \qquad\qquad r = 1, 2, \ldots, s.
\end{aligned} \tag{5.2}$$

If the constraint $\sum_{j=1}^{n} \lambda_j = 1$ is adjoined to (5.1), the fuzzy input-oriented BCC envelopment model is obtained and this added constraint introduces an additional variable, u_0, into the model (5.2), called fuzzy input-oriented BCC multiplier model, where these models are respectively formulated as follows:

$$\begin{aligned}
\min \quad & \theta_p \\
\text{s.t.} \quad & \sum_{j=1}^{n} \lambda_j \tilde{x}_{ij} \preceq \theta_p \tilde{x}_{ip}, \quad i = 1, 2, \ldots, m, \\
& \sum_{j=1}^{n} \lambda_j \tilde{y}_{rj} \succeq \tilde{y}_{rp}, \quad r = 1, 2, \ldots, s, \\
& \sum_{j=1}^{n} \lambda_j = 1, \\
& \lambda_j \geq 0, \qquad j = 1, 2, \ldots, n.
\end{aligned} \tag{5.3}$$

$$\begin{aligned}
\max \quad & \sum_{r=1}^{s} u_r \tilde{y}_{rp} + u_0 \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i \tilde{x}_{ip} = 1, \\
& \sum_{r=1}^{s} u_r \tilde{y}_{rj} - \sum_{i=1}^{m} v_i \tilde{x}_{ij} + u_0 \leq 0, \quad j = 1, 2, \ldots, n, \\
& \qquad\qquad\qquad\qquad\qquad\qquad i = 1, 2, \ldots, m, \\
& v_i \geq 0, \\
& u_r \geq 0, \qquad\qquad r = 1, 2, \ldots, s.
\end{aligned} \tag{5.4}$$

In what follows, several solution methods for solving FDEA models belonging to each category are explored.

5.2 The α–Level Approach

In this section, two procedures to measure the efficiencies of DMUs with fuzzy observations by applying alpha cuts are explored [2–4]. More studies in this area can be found in [5–10].

5.2.1 Kao and Liu's Approach

The basic idea behind the proposed approach by Kao and Liu [3] is to apply the alpha cuts and Zadeh's extension principle [11–14] to transform the fuzzy input-oriented BCC multiplier model (5.4) to a series of conventional crisp input-oriented BCC multiplier models.

Let $\left[\tilde{x}_{ij}\right]_\alpha = \left[\left(\tilde{x}_{ij}\right)_\alpha^L, \left(\tilde{x}_{ij}\right)_\alpha^U\right]$ and $\left[\tilde{y}_{rj}\right]_\alpha = \left[\left(\tilde{y}_{rj}\right)_\alpha^L, \left(\tilde{y}_{rj}\right)_\alpha^U\right]$ be the α-cuts of fuzzy input $\tilde{x}_{ij}$ and $\tilde{y}_{rj}$ fuzzy output, respectively. The membership function of fuzzy efficiency of DMU_p, denoted by $\tilde{E}_p$, can be defined as follows according to Zadeh's extension principle:

$$\mu_{\tilde{E}_p}(z) = \sup_{x,y} \min\left\{\mu_{\tilde{x}_{ij}}(x_{ij}), \mu_{\tilde{y}_{rj}}(y_{rj}), \forall i,j \middle| z = E_p(x,y)\right\} \tag{5.5}$$

where $E_p(x,y)$ is the efficiency of DMU_p in crisp input-oriented BCC multiplier model. The membership function of fuzzy efficiency of DMU_p defined in (5.5) can be approximated by calculating the lower bound $\left(\tilde{E}_p\right)_\alpha^L$ and the lower bound $\left(\tilde{E}_p\right)_\alpha^U$ of the fuzzy efficiency $\tilde{E}_p$ for a given α as follows:

$$\left(\tilde{E}_p\right)_\alpha^L = \min_{\substack{\left(\tilde{x}_{ij}\right)_\alpha^L \le x_{ij} \le \left(\tilde{x}_{ij}\right)_\alpha^U \\ \left(\tilde{y}_{rj}\right)_\alpha^L \le y_{rj} \le \left(\tilde{y}_{rj}\right)_\alpha^U, \\ \forall i,j}} \begin{cases} E_p = \max & \sum_{r=1}^{s} u_r y_{rp} + u_o \\ \text{s.t.} & \sum_{i=1}^{m} v_i x_{ip} = 1, \\ & \sum_{r=1}^{s} u_r y_{rj} - \sum_{i=1}^{m} v_i x_{ij} + u_o \le 0, \quad j = 1,2,\ldots,n, \\ & v_i \ge 0, \quad i = 1,2,\ldots,m, \\ & u_r \ge 0, \quad r = 1,2,\ldots,s. \end{cases} \tag{5.6}$$

$$\left(\tilde{E}_p\right)_\alpha^U = \max_{\substack{\left(\tilde{x}_{ij}\right)_\alpha^L \le x_{ij} \le \left(\tilde{x}_{ij}\right)_\alpha^U \\ \left(\tilde{y}_{rj}\right)_\alpha^L \le y_{rj} \le \left(\tilde{y}_{rj}\right)_\alpha^U, \\ \forall i,j}} \begin{cases} E_p = \max & \sum_{r=1}^{s} u_r y_{rp} + u_o \\ \text{s.t.} & \sum_{i=1}^{m} v_i x_{ip} = 1, \\ & \sum_{r=1}^{s} u_r y_{rj} - \sum_{i=1}^{m} v_i x_{ij} + u_o \le 0, \quad j = 1,2,\ldots,n, \\ & v_i \ge 0, \quad i = 1,2,\ldots,m, \\ & u_r \ge 0, \quad r = 1,2,\ldots,s. \end{cases} \tag{5.7}$$

The lower bound $\left(\tilde{E}_p\right)_\alpha^L$ is derived by setting $x_{ip} = \left(\tilde{x}_{ip}\right)_\alpha^U, y_{rp} = \left(\tilde{y}_{rp}\right)_\alpha^L$ and $x_{ij} = \left(\tilde{x}_{ij}\right)_\alpha^L, y_{rj} = \left(\tilde{y}_{rj}\right)_\alpha^U (j \neq p)$ in model (5.6). Therefore, two-level mathematical model (5.6) is simplified to the following one-level model:

$$\begin{aligned}
\left(\tilde{E}_p\right)_\alpha^L = \max \quad & \sum_{r=1}^{s} u_r \left(\tilde{y}_{rp}\right)_\alpha^L + u_o \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i \left(\tilde{x}_{ip}\right)_\alpha^U = 1, \\
& \sum_{r=1}^{s} u_r \left(\tilde{y}_{rp}\right)_\alpha^L - \sum_{i=1}^{m} v_i \left(\tilde{x}_{ip}\right)_\alpha^U + u_o \leq 0, \\
& \sum_{r=1}^{s} u_r \left(\tilde{y}_{rj}\right)_\alpha^U - \sum_{i=1}^{m} v_i \left(\tilde{x}_{ij}\right)_\alpha^L + u_o \leq 0, \quad j = 1, 2, \ldots, n, \ j \neq p, \\
& v_i \geq 0, \quad i = 1, 2, \ldots, m, \\
& u_r \geq 0, \quad r = 1, 2, \ldots, s.
\end{aligned} \tag{5.8}$$

On the contrary, the upper bound $\left(\tilde{E}_p\right)_\alpha^U$ is derived by setting $x_{ip} = \left(\tilde{x}_{ip}\right)_\alpha^L, y_{rp} = \left(\tilde{y}_{rp}\right)_\alpha^U$ and $x_{ij} = \left(\tilde{x}_{ij}\right)_\alpha^U, y_{rj} = \left(\tilde{y}_{rj}\right)_\alpha^L (j \neq p)$ in model (5.7). Therefore, two-level mathematical model (5.7) is simplified to the following one-level model:

$$\begin{aligned}
\left(\tilde{E}_p\right)_\alpha^U = \max \quad & \sum_{r=1}^{s} u_r \left(\tilde{y}_{rp}\right)_\alpha^U + u_o \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i \left(\tilde{x}_{ip}\right)_\alpha^L = 1, \\
& \sum_{r=1}^{s} u_r \left(\tilde{y}_{rp}\right)_\alpha^U - \sum_{i=1}^{m} v_i \left(\tilde{x}_{ip}\right)_\alpha^L + u_o \leq 0, \\
& \sum_{r=1}^{s} u_r \left(\tilde{y}_{rj}\right)_\alpha^L - \sum_{i=1}^{m} v_i \left(\tilde{x}_{ij}\right)_\alpha^U + u_o \leq 0, \quad j = 1, 2, \ldots, n, \ j \neq p, \\
& v_i \geq 0, \quad i = 1, 2, \ldots, m, \\
& u_r \geq 0, \quad r = 1, 2, \ldots, s.
\end{aligned} \tag{5.9}$$

In sum, the α-cut of the fuzzy efficiency for DMU_p, $\left[\tilde{E}_p\right]_\alpha = \left[\left(\tilde{E}_p\right)_\alpha^L, \left(\tilde{E}_p\right)_\alpha^U\right]$, is constructed by solving the crisp linear programming models (5.8) and (5.9). Kao and Liu [3] after calculating the fuzzy efficiency scores of different DMUs have used two existing approaches [15, 16] of ranking fuzzy numbers methods to order of the DMUs.

It is worth noting that model (5.8) is simplified to the following model for triangular fuzzy input data $\tilde{x}_{ij} = \left(x_{ij,1}, x_{ij,2}, x_{ij,3}\right)$ and triangular fuzzy output data $\tilde{y}_{rj} = \left(y_{rj,1}, y_{rj,2}, y_{rj,3}\right)$:

$$
\begin{aligned}
\left(\tilde{E}_p\right)_{\alpha}^{L} = \max \quad & \sum_{r=1}^{s} u_r\left(y_{rp,1} + (y_{rp,2} - y_{rp,1})\alpha\right) + u_o \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i\left(x_{ip,3} - (x_{ip,3} - x_{ip,2})\alpha\right) = 1, \\
& \sum_{r=1}^{s} u_r\left(y_{rp,1} + (y_{rp,2} - y_{rp,1})\alpha\right) \\
& - \sum_{i=1}^{m} v_i\left(x_{ip,3} - (x_{ip,3} - x_{ip,2})\alpha\right) + u_o \le 0, \\
& \sum_{r=1}^{s} u_r\left(y_{rj,3} - (y_{rj,3} - y_{rj,2})\alpha\right) \\
& - \sum_{i=1}^{m} v_i\left(x_{ij,1} + (x_{ij,2} - x_{ij,1})\alpha\right) + u_o \le 0, \quad j = 1, 2, \ldots, n, j \ne p, \\
& v_i \ge 0, \quad i = 1, 2, \ldots, m, \\
& u_r \ge 0, \quad r = 1, 2, \ldots, s.
\end{aligned}
\tag{5.10}
$$

Assume

$$
\begin{aligned}
\dot{x}_{ip} &= \left(x_{ip,3} - (x_{ip,3} - x_{ip,2})\alpha\right), && \forall i, j = p \\
\dot{y}_{rp} &= \left(y_{rp,1} + (y_{rp,2} - y_{rp,1})\alpha\right), && \forall r, j = p \\
\hat{x}_{ij} &= \left(x_{ij,1} + (x_{ij,2} - x_{ij,1})\alpha\right), && \forall i, \forall j \ne p \\
\hat{y}_{rj} &= \left(y_{rj,3} - (y_{rj,3} - y_{rj,2})\alpha\right), && \forall r, \forall j \ne p
\end{aligned}
\tag{5.11}
$$

In matrix form, the problem (5.10) can be stated as follows:

$$
\begin{aligned}
\max \quad & \left[0, \ldots, 0, \hat{y}_{1p}, \ldots, \hat{y}_{sp}, 1, -1\right]\left[v_1, \ldots, v_m, u_1, \ldots, u_s, u_0^+, u_0^-\right]^{\mathrm{t}} \\
\text{s.t.} \quad & \left[\begin{array}{ccc:ccc:ccc}
-\dot{x}_{11} & \cdots & -\dot{x}_{m1} & \dot{y}_{11} & \cdots & \dot{y}_{s1} & 1 & \cdots & -1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
-\hat{x}_{1p} & \cdots & -\hat{x}_{1p} & \hat{y}_{1p} & \cdots & \hat{y}_{1p} & 1 & \cdots & -1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
-\dot{x}_{1n} & \cdots & -\dot{x}_{mn} & \dot{y}_{1n} & \cdots & \dot{y}_{sn} & 1 & \cdots & -1 \\
\hdashline
\hat{x}_{1p} & \cdots & \hat{x}_{mp} & 0 & \cdots & 0 & 0 & \cdots & 0
\end{array}\right]
\left[\begin{array}{c}
v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \\ \hdashline u_0^+ \\ u_0^+
\end{array}\right]
\begin{array}{l}
\le_n \\ =_1
\end{array}
\left[\begin{array}{c}
0_{n\times 1} \\ \hdashline 1
\end{array}\right]
\end{aligned}
\tag{5.12}
$$

In a similar way, model (5.9) is simplified to the following model for triangular fuzzy input data $\tilde{x}_{ij} = (x_{ij,1}, x_{ij,2}, x_{ij,3})$ and triangular fuzzy output data $\tilde{y}_{rj} = (y_{rj,1}, y_{rj,2}, y_{rj,3})$:

$$
\begin{aligned}
\left(\tilde{E}_p\right)_\alpha^U = \max \quad & \sum_{r=1}^{s} u_r\left(y_{rp,3} - (y_{rp,3} - y_{rp,2})\alpha\right) + u_o \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i\left(x_{ip,1} + (x_{ip,2} - x_{ip,1})\alpha\right) = 1, \\
& \sum_{r=1}^{s} u_r\left(y_{rp,3} - (y_{rp,3} - y_{rp,2})\alpha\right) \\
& - \sum_{i=1}^{m} v_i\left(x_{ip,1} + (x_{ip,2} - x_{ip,1})\alpha\right) + u_o \le 0, \\
& \sum_{r=1}^{s} u_r\left(y_{rj,1} + (y_{rj,2} - y_{rj,1})\alpha\right) \\
& - \sum_{i=1}^{m} v_i\left(x_{ij,3} - (x_{ij,3} - x_{ij,2})\alpha\right) + u_o \le 0, \quad j = 1, 2, \ldots, n, j \ne p, \\
& v_i \ge 0, \quad i = 1, 2, \ldots, m, \\
& u_r \ge 0, \quad r = 1, 2, \ldots, s.
\end{aligned}
\tag{5.13}
$$

Assume

$$
\begin{aligned}
\hat{x}_{ip} &= \left(x_{ip,3} - (x_{ip,3} - x_{ip,2})\alpha\right), && \forall i, j = p \\
\hat{y}_{rp} &= \left(y_{rp,1} + (y_{rp,2} - y_{rp,1})\alpha\right), && \forall r, j = p \\
\dot{x}_{ij} &= \left(x_{ij,1} + (x_{ij,2} - x_{ij,1})\alpha\right), && \forall i, \forall j \ne p \\
\dot{y}_{rj} &= \left(y_{rj,3} - (y_{rj,3} - y_{rj,2})\alpha\right), && \forall r, \forall j \ne p
\end{aligned}
\tag{5.14}
$$

In matrix form, the problem (5.13) can be stated as follows:

$$
\text{s.t.} \quad
\left[\begin{array}{ccc:ccc:ccc}
-\hat{x}_{11} & \cdots & -\hat{x}_{m1} & \hat{y}_{11} & \cdots & \hat{y}_{s1} & 1 & \cdots & -1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
-\dot{x}_{1p} & \cdots & -\dot{x}_{1p} & \dot{y}_{1p} & \cdots & \dot{y}_{1p} & 1 & \cdots & -1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
-\hat{x}_{1n} & \cdots & -\hat{x}_{mn} & \hat{y}_{1n} & \cdots & \hat{y}_{sn} & 1 & \cdots & -1 \\
\hdashline
\dot{x}_{1p} & \cdots & \dot{x}_{mp} & 0 & \cdots & 0 & 0 & \cdots & 0
\end{array}\right]
\left[\begin{array}{c}
v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \\ \hdashline u_0^+ \\ u_0^+
\end{array}\right]
\begin{array}{l}
\le_n \\ =_1
\end{array}
\left[\begin{array}{c}
0_{n\times 1} \\ \hdashline 1
\end{array}\right]
\tag{5.15}
$$

The R code for model (5.12) is as follows.

```
# required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
# reading data
df1 = data.frame(read_excel(path = "data7.xlsx", sheet="1"),
check.names = F, fix.empty.names = TRUE)
df2 = data.frame(read_excel(path = "data8.xlsx", sheet="1"),
check.names = F, fix.empty.names = TRUE)
df3 = data.frame(read_excel(path = "data9.xlsx", sheet="1"),
check.names = F, fix.empty.names = TRUE)
# number of elements and parameters
alpha=0
m=4
s = ncol(df1)-m
N = nrow(df1)
# defining right hand side and directions
f.rhs = c(1, rep(0,N))
f.dir = c("=", rep("<=",N))
#one of constraints' coefficient
con2=  cbind((df3[,(m+1):(m+s)]-(df3[,(m+1):(m+s)]-df2[,(m+1):(m+s)])
*alpha), -(df1[,1:m]+(df2[,1:m]-df1[,1:m])*alpha), 1, -1)
for (j in 1:N) {
# defining objective and matrix of constraints' coefficient
f.obj  =  c(((df1[j,(m+1):(m+s)])+((df2[j,(m+1):(m+s)])-(df1[j,(m+1):(m
+s)]))*alpha), rep(0,m), 1, -1)
con1=  c(rep(0,s),  (as.numeric(df3[j,1:m])-(as.numeric(df3[j,1:m])-as.
numeric(df2[j,1:m]))*alpha), 0, 0)
con2[j,]= cbind(((df1[j,(m+1):(m+s)])+((df2[j,(m+1):(m+s)])-(df1[j,(m+1):
(m+s)]))*alpha),-(df3[j,1:m]-(df3[j,1:m]-df2[j,1:m])*alpha),1,-1)
f.con = rbind(con1, con2)
#solving model
result = lp('max', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
multiplier = result$solution
u0 = multiplier[s+m+1]-multipliers[s+m+2]
if (j==1) {
weight = c(multiplier[seq(1,s+m)],u0)
effvr = result$objval
} else {
weight = rbind(weight,c(multiplier[seq(1,s+m)],u0))
effvr = rbind(effvr, result$objval)
}
```

```
}
EEs = data.frame(effvr, weight)
colnames(EEs) = c('effic', rep('v', m), rep('u', s), 'u0')
WriteXLS(EEs,"Low.xls", row.names = TRUE, col.names = TRUE)
```

In a similar way, the R code for model (5.15) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data7.xlsx", sheet="1"),
check.names = F, fix.empty.names = TRUE)
df2 = data.frame(read_excel(path = "data8.xlsx", sheet="1"),
check.names = F, fix.empty.names = TRUE)
df3 = data.frame(read_excel(path = "data9.xlsx", sheet="1"),
check.names = F, fix.empty.names = TRUE)
# parameters and number of elements
alpha=0.75
m=4
s = ncol(df1)-m
N = nrow(df1)
# defining right hand side and direction
f.rhs = c(1, rep(0,N))
f.dir = c("=", rep("<=", N))
#one of the constraints' coefficient
con2=  cbind((df3[,(m+1):(m+s)]-(df3[,(m+1):(m+s)]-df2[,(m+1):(m+s)])
*alpha), -(df1[,1:m]+(df2[,1:m]-df1[,1:m])*alpha), 1, -1)
for (j in 1:N) {
#defining objective and and matrix of constraints' coefficient
f.obj = c(((df3[j,(m+1):(m+s)])-((df3[j,(m+1):(m+s)])-(df2[j,(m+1):(m+s)]))
*alpha), rep(0,m),1,-1)
con1=  c(rep(0,s),  (as.numeric(df1[j,1:m])+(as.numeric(df2[j,1:m])-as.
numeric(df1[j,1:m]))*alpha), 0, 0)
con2[j,]=  cbind(((df3[j,(m+1):(m+s)])-((df3[j,(m+1):(m+s)])-(df2[j,(m+1):
(m+s)]))*alpha),-(df1[j,1:m]+(df2[j,1:m]-df1[j,1:m])*alpha),1,-1)
f.con = rbind(con1, con2)
#solving model
result = lp('max', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
```

```
multiplier = result$solution
u0 = multiplier[s+m+1]-multipliers[s+m+2]
if (j==1) {
weight = c(multiplier[seq(1,s+m)],u0)
effvr = result$objval
} else {
weight = rbind(weight,c(multiplier[seq(1,s+m)],u0))
effvr = rbind(effvr, result$objval) } }
EEs = data.frame(effvr, weight)
colnames(EEs) = c('effic', rep('v', m), rep('u', s), 'u0'); EEsWriteXLS
(EEs,"up.xls", row.names = TRUE, col.names = TRUE)
```

Example 5.1 Consider a fuzzy DEA efficiency evaluation problem with five DMUs, each DMU with two fuzzy inputs and two fuzzy outputs. The data set is taken from Guo and Tanaka [17] and is shown in Table 5.1.

The range of efficiency scores of DMUs at different possibility levels derived by applying the α-cuts approach is given in Table 5.2.

Consider DMU C in this example. For possibility level $\alpha = 0.75$, the range of the efficiency score is [0.83, 0.97], indicating that DMU C's efficiency score will never exceed 0.97 or fall below 0.83.

Table 5.1 Five DMUs with two fuzzy inputs and two fuzzy outputs

DMU	A	B	C	D	E
Input 1	(3.5, 4.0, 4.5)	(2.9, 2.9, 2.9)	(4.4, 4.9, 5.4)	(3.4, 4.1, 4.8)	(5.9, 6.5, 7.1)
Input 2	(1.9, 2.1, 2.3)	(1.4, 1.5, 1.6)	(2.2, 2.6, 3.0)	(2.2, 2.3, 2.4)	(3.6, 4.1, 4.6)
Output 1	(2.4, 2.6, 2.8)	(2.2, 2.2, 2.2)	(2.7, 3.2, 3.7)	(2.5, 2.9, 3.3)	(4.4, 5.1, 5.8)
Output 2	(3.8, 4.1, 4.4)	(3.3, 3.5, 3.7)	(4.3, 5.1, 5.9)	(5.5, 5.7, 5.9)	(6.5, 7.4, 8.3)

Table 5.2 The alpha cuts of the fuzzy efficiencies

α	$\left[(\tilde{E}_A)_\alpha^U, (\tilde{E}_A)_\alpha^U\right]$	$\left[(\tilde{E}_B)_\alpha^U, (\tilde{E}_B)_\alpha^U\right]$	$\left[(\tilde{E}_C)_\alpha^U, (\tilde{E}_C)_\alpha^U\right]$	$\left[(\tilde{E}_D)_\alpha^U, (\tilde{E}_D)_\alpha^U\right]$	$\left[(\tilde{E}_E)_\alpha^U, (\tilde{E}_E)_\alpha^U\right]$
0.0	(0.67, 0.94)	(1.00, 1.00)	(0.62, 1.00)	(1.00, 1.00)	(1.00, 1.00)
0.25	(0.71, 0.93)	(1.00, 1.00)	(0.68, 1.00)	(1.00, 1.00)	(1.00, 1.00)
0.5	(0.76, 0.91)	(1.00, 1.00)	(0.75, 1.00)	(1.00, 1.00)	(1.00, 1.00)
0.75	(0.82, 0.90)	(1.00, 1.00)	(0.83, 0.97)	(1.00, 1.00)	(1.00, 1.00)
1.0	(0.89, 0.89)	(1.00, 1.00)	(0.94, 0.94)	(1.00, 1.00)	(1.00, 1.00)

5.2.2 Saati et al.'s Approach

The basic idea behind the proposed approach by Saati et al. [4] is to apply the alpha cuts to transform the fuzzy input-oriented CCR multiplier model (5.2) to a crisp linear programming problem.

Let $\tilde{x}_{ij} = \left(x_{ij,1}, x_{ij,2}, x_{ij,3}\right)$ and $\tilde{y}_{rj} = \left(y_{rj,1}, y_{rj,2}, y_{rj,3}\right)$ be triangular fuzzy input data and triangular fuzzy output data, respectively. The α-cuts of fuzzy input data and fuzzy output data are given as $\left[\tilde{x}_{ij}\right]_\alpha = \left[x_{ij,1} + (x_{ij,2} - x_{ij,1})\alpha, x_{ij,3} - (x_{ij,3} - x_{ij,2})\alpha\right]$ and $\left[\tilde{y}_{rj}\right]_\alpha = \left[y_{rj,1} + (y_{rj,2} - y_{rj,1})\alpha, y_{rj,3} - (y_{rj,3} - y_{rj,2})\alpha\right]$, respectively. In this case, model (5.2) is converted into the following interval programming problem for a given α:

$$
\begin{aligned}
\tilde{E}_p = \max \quad & \sum_{r=1}^{s} u_r \left[y_{rp,1} + (y_{rp,2} - y_{rp,1})\alpha, y_{rp,3} - (y_{rp,3} - y_{rp,2})\alpha\right] \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i \left[x_{ip,1} + (x_{ip,2} - x_{ij,1})\alpha, x_{ip,3} - (x_{ip,3} - x_{ip,2})\alpha\right] = [1,1], \\
& \sum_{r=1}^{s} u_r \left[\tilde{y}_{rj}\right]_\alpha = \left[y_{rj,1} + (y_{rj,2} - y_{rj,1})\alpha, y_{rj,3} - (y_{rj,3} - y_{rj,2})\alpha\right] \\
& \quad - \sum_{i=1}^{m} v_i \left[x_{ij,1} + (x_{ij,2} - x_{ij,1})\alpha, x_{ij,3} - (x_{ij,3} - x_{ij,2})\alpha\right] \le [0,0], \quad \forall j \\
& v_i \ge 0, \quad i = 1, 2, \ldots, m, \\
& u_r \ge 0, \quad r = 1, 2, \ldots, s.
\end{aligned}
\tag{5.16}
$$

Saati et al. [4] converted the interval programming model (5.16) into a nonlinear programming problem by first defining new variables $\hat{x}_{ij} \in \left[\tilde{x}_{ij}\right]_\alpha$ and $\hat{y}_{rj} \in \left[\tilde{y}_{rj}\right]_\alpha$; and then substituting these variables in model (5.16). Finally, the resulting nonlinear programming problem is converted into the following linear programming problem by defining $\bar{x}_{ij} = v_i \hat{x}_{ij}$, $\bar{y}_{rj} = u_r \hat{y}_{rj}$:

$$
\begin{aligned}
E_p^\alpha = \max \quad & \sum_{r=1}^{s} u_r \bar{y}_{rp} \\
\text{s.t.} \quad & \sum_{i=1}^{m} \bar{x}_{ip} = 1, \\
& \sum_{r=1}^{s} \bar{y}_{rj} - \sum_{i=1}^{m} \bar{x}_{ij} \le 0, && \forall j \\
& v_i \left[x_{ij,1} + (x_{ij,2} - x_{ij,1})\alpha\right] \le \bar{x}_{ij} \le v_i \left[x_{ij,3} - (x_{ij,3} - x_{ij,2})\alpha\right], && \forall i, j \\
& u_r \left[y_{rj,1} + (y_{rj,2} - y_{rj,1})\alpha\right] \le \bar{y}_{rj} \le u_r \left[y_{rj,3} - (y_{rj,2} - y_{rj,3})\alpha\right], && \forall r, j \\
& v_i \ge 0, \quad i = 1, 2, \ldots, m, \\
& u_r \ge 0, \quad r = 1, 2, \ldots, s.
\end{aligned}
\tag{5.17}
$$

Definition 5.1 DMU_p is said to be efficient at given $\alpha \in (0, 1]$, if $E_p^{\alpha} = 1$.

Assume

$$\begin{aligned}
\dot{x}_{ij} &= \left[x_{ij,1} + (x_{ij,2} - x_{ij,1})\alpha\right] \quad \forall i, \forall j \\
\ddot{x}_{ij} &= \left[x_{ij,3} - (x_{ij,3} - x_{ij,2})\alpha\right] \quad \forall i, \forall j \\
\dot{y}_{rj} &= \left[y_{rj,1} + (y_{rj,2} - y_{rj,1})\alpha\right] \quad \forall r, \forall j \\
\ddot{y}_{rj} &= \left[y_{rj,3} - (y_{rj,2} - y_{rj,3})\alpha\right] \quad \forall r, \forall j
\end{aligned} \tag{5.18}$$

In matrix form, the problem (5.18) can be stated as follows:

$$\begin{aligned}
&\max \quad [1, 0, 0, 0, 0, 0]\left[\vec{Y}p, \vec{X}p, \vec{Yj}, \vec{X}j, \vec{V}, \vec{U}\right]^t \\
&\text{s.t.} \quad \begin{bmatrix} 0 & 1_{1m} & 0 & 0 & 0 & 0 \\ 0 & 0 & -I_n & I_n & 0 & 0 \\ 0 & 0 & 0 & -1_{nm} & \dot{x}_{nm} & 0 \\ 0 & 0 & 0 & 1_{nm} & \ddot{x}_{nm} & 0 \\ 0 & 0 & -1 & 0 & 0 & \dot{y}_{ns} \\ 0 & 0 & 1 & 0 & 0 & \ddot{y}_{ns} \end{bmatrix} \begin{bmatrix} \vec{Y}p \\ \vec{X}p \\ \vec{Y}j \\ \vec{X}j \\ \vec{V} \\ \vec{U} \end{bmatrix} \begin{matrix} = \\ \leq_n \\ \leq_n \\ \leq_n \\ \leq_n \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
&U \geq 0, V \geq 0
\end{aligned} \tag{5.19}$$

The R code of model (5.19) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = (data.frame(read_excel(path = "data7.xlsx", sheet="1")))
df2 = (data.frame(read_excel(path = "data8.xlsx", sheet="1")))
df3 = (data.frame(read_excel(path = "data9.xlsx", sheet="1")))
IN = (data.frame(read_excel(path = "I10.xlsx", sheet="1")))
ON = (data.frame(read_excel(path = "O10.xlsx", sheet="1")))
#parameters and number of elements
alpha=1
m=2
s=2
N = nrow(df1)
#defining directions
f.dir = c("=", rep("<=",9*N))
for (j in 1:N) {
```

```
#defining right hand side, objective, and matrix of coefficient
f.rhs = c(1, rep(0,N), rep(0,8*N))
f.obj = c(as.numeric(ON[j,]), rep(0,(N+m+s)))
con1= c(rep(0,N), as.numeric(IN[j,]), rep(0,(s+m)))
con2= cbind(diag(N), -diag(N), matrix(0,N,(m+s)))
con3= cbind(matrix(0,N,N), -diag(N), (df1[,1]+(df2[,1]-df1[,1])*alpha),
(matrix(0,N,s)))
con4= cbind(matrix(0,N,N), diag(N), -(df3[,1]-(df3[,1]-df2[,1])*alpha),
(matrix(0,N,s)))
con5= cbind(-diag(N), (matrix(0,N,N+m)), (df1[,5]+(df2[,5]-df1[,5])*alpha))
con6= cbind(diag(N), (matrix(0,N,N+m)), -(df3[,5]-(df3[,5]-df2[,5])*alpha))
a.con1=rbind(con3, con4)
a.con2=rbind(con5, con6)
f.con = rbind(con1, con2, a.con1, a.con2)
#solving model
results = lp ("max",f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weights = results$solution
effcrs = results$objval
} else {
weights = rbind(weights, results$solution)
effcrs = rbind(effcrs, results$objval)
}
}
ESs = data.frame(effcrs, weights)
colnames(EEs) = c('effi', "weights")
WriteXLS(EEs,"4.11.xls", row.names = F, col.names = F)
```

Example 5.2 Consider the data given in Table 5.1. Fuzzy efficiencies of DMUs with the Saati et al.'s method [4] with different α values are listed in Table 5.3. Considering $\alpha = 0.75$, DMUs B, D and E are efficient and DMUs A and C are inefficient.

Table 5.3 The efficiency based on model (5.17)

α	E_A^α	E_B^α	E_C^α	E_D^α	E_E^α
0.0	1.00	1.00	1.00	1.00	1.00
0.25	0.98	1.00	1.00	1.00	1.00
0.5	0.96	1.00	1.00	1.00	1.00
0.75	0.91	1.00	0.93	1.00	1.00
1.0	0.85	1.00	0.86	1.00	1.00

5.3 The Fuzzy Ranking Approach

In this section, three procedures to measure the fuzzy efficiencies of DMUs with fuzzy input data and fuzzy output data by using some fuzzy ranking methods are explored [17–19]. More studies on this area can be found in [20–24].

5.3.1 Guo and Tanaka's Approach

Guo and Tanaka [17] initially developed fuzzy ranking approach for measuring fuzzy efficiencies of DMUs with fuzzy data based on the following fuzzy input-oriented CCR multiplier model:

$$\begin{aligned} \max \quad & \sum_{r=1}^{s} u_r \tilde{y}_{rp} \\ \text{s.t.} \quad & \sum_{i=1}^{m} v_i \tilde{x}_{ip} \simeq \tilde{1}, \\ & \sum_{r=1}^{s} u_r \tilde{y}_{rj} \preceq \sum_{i=1}^{m} v_i \tilde{x}_{ij}, \quad j = 1, 2, \ldots, n, \\ & v_i \geq 0, \qquad\qquad\qquad i = 1, 2, \ldots, m, \\ & u_r \geq 0, \qquad\qquad\qquad r = 1, 2, \ldots, s. \end{aligned} \tag{5.20}$$

Let $\tilde{x}_{ij} = \left(x_{ij} - t_{ij}, x_{ij}, x_{ij} + t_{ij}\right)$ and $\tilde{y}_{rj} = \left(y_{rj} - k_{rj}, y_{rj}, y_{rj} + k_{ij}\right)$ be symmetric triangular fuzzy input data and fuzzy output data, respectively, where x_{ij} and y_{rj} are the centers; and t_{ij} and k_{ij} are the spreads of the respective input and output. Moreover, the fuzzy unity given in the first constraint of model (5.20) is a symmetric triangular fuzzy number as $\tilde{1} = (1 - e, 1, 1 + e)$.

Guo and Tanaka [17] defined the concepts of "almost equal", "almost less than" and "maximizing a fuzzy variable" for solving FDEA model (5.20).

Definition 5.2 Given two symmetric triangular fuzzy numbers $\tilde{A}_1 = (a_1 - t_1, a_1, a_1 + t_1)$ and $\tilde{A}_2 = (a_2 - t_2, a_2, a_2 + t_2)$ we say $\tilde{A}_1 \preceq \tilde{A}_2$ if $a_1 - (1 - \alpha)t_1 \leq a_2 - (1 - \alpha)t_2$ and $a_1 + (1 - \alpha)t_1 \leq a_2 + (1 - \alpha)t_2$, where $\alpha \in [0, 1]$ is a predetermined possibility level by decision-makers.

Remark 5.1 Refereeing to Definition 5.2, maximization $\sum_{r=1}^{s} u_r \tilde{y}_{rp}$ can be expressed as maximization $\sum_{r=1}^{s} u_r \left[\tilde{y}_{rp}\right]_{\alpha} = \left[\sum_{r=1}^{s} u_r \left(y_{rp} - (1 - \alpha)k_{rp}\right), \sum_{r=1}^{s} u_r \left(y_{rp} + (1 - \alpha)k_{rp}\right)\right]$. From the pessimistic opinion of maximizing $\sum_{r=1}^{s} u_r \tilde{y}_{rp}$, it is reduced to maximization $\sum_{r=1}^{s} u_r \left(y_{rp} - (1 - \alpha)k_{rp}\right)$.

Remark 5.2 Regarding Definition 5.2, the problem of finding out $v = (v_1, \ldots, v_m)$ such that $\sum_{i=1}^{m} v_i \tilde{x}_{ip} \simeq \tilde{1}$, is reduced to the following optimization problem:

$$\begin{aligned}
\max \quad & \sum_{i=1}^{m} v_i t_{ip} \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ip} - (1-\alpha) v_i t_{ip} = 1 - (1-\alpha) e \\
& \sum_{i=1}^{m} v_i x_{ip} + (1-\alpha) v_i t_{ip} \le 1 + (1-\alpha) e \\
& v_i \ge 0, \qquad\qquad i = 1, 2, \ldots, m.
\end{aligned} \tag{5.21}$$

Regarding Definition 5.2, Remarks 5.1 and 5.2, the FDEA model (5.20) is converted to the following optimization problem:

$$\begin{aligned}
\max \quad & \sum_{r=1}^{s} u_r \left(y_{rj} - (1-\alpha) k_{rj} \right) \\
\text{s.t.} \max \quad & \sum_{i=1}^{m} v_i t_{ip} \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ip} - (1-\alpha) v_i t_{ip} = 1 - (1-\alpha) e \\
& \sum_{i=1}^{m} v_i x_{ip} + (1-\alpha) v_i t_{ip} \le 1 + (1-\alpha) e \\
& v_i \ge 0, \qquad\qquad i = 1, 2, \ldots, m, \\
& \sum_{r=1}^{s} u_r \left(y_{rj} - (1-\alpha) k_{rj} \right) \le \sum_{i=1}^{m} v_i \left(x_{ij} - (1-\alpha) t_{ij} \right), \quad j = 1, 2, \ldots, n, \\
& \sum_{r=1}^{s} u_r \left(y_{rj} + (1-\alpha) k_{rj} \right) \le \sum_{i=1}^{m} v_i \left(x_{ij} + (1-\alpha) t_{ij} \right), \quad j = 1, 2, \ldots, n, \\
& u_r \ge 0, \qquad\qquad r = 1, 2, \ldots, s.
\end{aligned} \tag{5.22}$$

Assume the optimal value of the problem is g_p when e is taken as $e = \max\limits_{j=1,\ldots,n} \left\{ \max\limits_{i=1,\ldots,m} \left\{ \dfrac{t_{ij}}{x_{ij}} \right\} \right\}$. In this case, the optimization problem (5.22) can be reformulated as the following linear programming problem:

$$\begin{aligned}
\max \quad & \sum_{r=1}^{s} u_r \left(y_{rj} - (1-\alpha) k_{rj} \right) \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i t_{ip} \ge g_p \\
& \sum_{i=1}^{m} v_i x_{ip} - (1-\alpha) v_i t_{ip} = 1 - (1-\alpha) e \\
& \sum_{i=1}^{m} v_i x_{ip} + (1-\alpha) v_i t_{ip} \le 1 + (1-\alpha) e \\
& \sum_{r=1}^{s} u_r \left(y_{rj} - (1-\alpha) k_{rj} \right) \le \sum_{i=1}^{m} v_i \left(x_{ij} - (1-\alpha) t_{ij} \right), \quad j = 1, 2, \ldots, n, \\
& \sum_{r=1}^{s} u_r \left(y_{rj} + (1-\alpha) k_{rj} \right) \le \sum_{i=1}^{m} v_i \left(x_{ij} + (1-\alpha) t_{ij} \right), \quad j = 1, 2, \ldots, n, \\
& v_i \ge 0, \qquad\qquad i = 1, 2, \ldots, m, \\
& u_r \ge 0, \qquad\qquad r = 1, 2, \ldots, s.
\end{aligned} \tag{5.23}$$

Assume

$$\begin{array}{ll}
\dot{y}_{rj} = \left(y_{rj} - (1-\alpha)k_{rj}\right)\forall r, \forall j & \dot{x}_{ip} = \left(v_i x_{ip} - (1-\alpha)t_{ip}\right)\forall i, j = p \\
\dot{x}_{ij} = \left(x_{ij} - (1-\alpha)t_{ij}\right)\forall i, \forall j & \dot{y}_{rp} = \left(y_{rp} - (1-\alpha)k_{rp}\right)\forall r, j = p \\
\ddot{y}_{rj} = \left(y_{rj} + (1-\alpha)k_{rj}\right)\forall r, \forall j & \ddot{x}_{ip} = \left(v_i x_{ip} - (1-\alpha)t_{ip}\right)\forall i, j = p \\
\ddot{x}_{ij} = \left(x_{ij} + (1-\alpha)t_{ij}\right)\forall i, \forall j & \ddot{y}_{rp} = \left(y_{rp} - (1-\alpha)k_{rp}\right)\forall r, j = p \\
1^{+} = 1 + (1-e)\alpha & 1^{-} = 1 - (1-e)\alpha
\end{array} \tag{5.24}$$

In matrix form, the problem (5.23) can be stated as follows:

$$\begin{aligned}
&\max \quad \left[0, \ldots, 0, y_{1p}, \ldots, y_{sp}\right]\left[v_1, \ldots, v_m, u_1, \ldots, u_s\right]^{\mathrm{t}} \\
&s.t. \quad \left[\begin{array}{cccc:ccc}
t_{1p} & \cdots & t_{mp} & 0 & \cdots & 0 \\
\dot{x}_{1p} & \cdots & \dot{x}_{1p} & \vdots & & \vdots \\
\ddot{x}_{1p} & \cdots & \ddot{x}_{mp} & 0 & \cdots & 0 \\
\hdashline
-\dot{x}_{11} & \cdots & -\dot{x}_{m1} & \dot{y}_{11} & \cdots & \dot{y}_{s1} \\
\vdots & & \vdots & \vdots & & \vdots \\
-\dot{x}_{1n} & \cdots & -\dot{x}_{mn} & \dot{y}_{1n} & \cdots & \dot{y}_{sn} \\
-\ddot{x}_{11} & \cdots & -\ddot{x}_{m1} & \ddot{y}_{11} & \cdots & \ddot{y}_{s1} \\
\vdots & & \vdots & \vdots & & \vdots \\
-\ddot{x}_{1n} & \cdots & -\ddot{x}_{mn} & \ddot{y}_{1n} & \cdots & \ddot{y}_{sn} \\
\hdashline
\end{array}\right]
\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \end{array}\right]
\begin{array}{c} \geq \\ \hdashline = \\ \leq \\ \leq \\ \leq \end{array}
\left[\begin{array}{c} g_p \\ \hdashline 1^{-} \\ 1^{+} \\ 0 \\ 0 \end{array}\right] \\
&v_i \geq 0, \quad i = 1, 2, \ldots, m, \\
&u_r \geq 0, \quad r = 1, 2, \ldots, s.
\end{aligned} \tag{5.25}$$

Definition 5.3 The fuzzy efficiency of DMU_p, $\tilde{E}_p^* = (E_p^{*l}, E_p^{*m}, E_p^{*u})$, is defined as follows:

$$\begin{aligned}
\tilde{E}_p^* &= (E_p^{*l}, E_p^{*m}, E_p^{*u}) \\
&= \left(\frac{\sum_{r=1}^{s} u_r^* \left(y_{rp} - (1-\alpha)k_{rp}\right)}{\sum_{i=1}^{m} v_i^* \left(x_{ip} + (1-\alpha)t_{ip}\right)}, \frac{\sum_{r=1}^{s} u_r^* y_{rp}}{\sum_{i=1}^{m} v_i^* x_{ip}}, \frac{\sum_{r=1}^{s} u_r^* \left(y_{rp} + (1-\alpha)k_{rp}\right)}{\sum_{i=1}^{m} v_i^* \left(x_{ip} - (1-\alpha)t_{ip}\right)}\right)
\end{aligned}$$

Definition 5.4 DMU_p with $E_p^{*u} \geq 1$ for the α possibility level is called an $\alpha-$possibilistic D efficient DMU (PD DMU); otherwise it is called an α-possibilistic D inefficient DMU (PDI DMU). The set of all PD DMUs is called the α-possibilistic non-dominated set, denoted as S_α.

The R code for model (5.23) is as follows. In this code, the value of "*e*", defined as a parameter, is calculated according to the given formula. Also, as fuzzy data are symmetric triangular fuzzy ones, thus "*t*" and "*k*" are defined with the same name in

the code. Moreover, in this code for each possibility level α, the model (5.21) is solved and then its optimal solution, denoted by "g", is used in model (5.23).

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df = data.frame(read_excel(path = "data1new.xlsx", sheet = "1"))
df1 = data.frame(read_excel(path = "data1L.xlsx", sheet = "1"))
df2 = data.frame(read_excel(path = "123.xlsx", sheet = "1"))
t=data.frame(df1[1:2])
k=data.frame(df1[3:4])
#number of elements and parameters
m=2
s = ncol(df)-m
N = nrow(df)
alpha=0.75;
e=0.2;
#defining directions
f.dir = c(">=", "=", "<=", rep("<=",(2*N)), rep(">=",(m+s)))
for (j in 1:N) {
#defining right hand side and objective
f.rhs=c(as.numeric(df2[j,1]), 1-(1-alpha)*e, 1+(1-alpha)*e, rep(0,(2*N)),
rep(0.00001,m), rep(0.00001, s))
f.ob=c(rep(0,m),(as.numeric(df[j,(m+1):(m+s)])-(1-alpha)*(as.numeric(df1
[j,(m+1):(m+s)]))))
#defining matrix of coefficient
con1= c(as.numeric(df1[j,1:m]), rep(0,s))
con2= c(as.numeric(df[j,1:m])-(1-alpha)*as.numeric(df1[j,1:m]), rep(0,s))
con3= c(as.numeric(df[j,1:m])+(1-alpha)*as.numeric(df1[j,1:m]), rep(0,s))
con4=cbind(-(df[,1:m]-(1-alpha)*df1[,1:m]),df[,(m+1):(m+s)]-(1-alpha)
*df1[,(m+1):(m+s)])
con5=cbind(-(df[,1:m]+(1-alpha)*df1[,1:m]),df[,(m+1):(m+s)]+(1-alpha)
*df1[,(m+1):(m+s)])
con6= cbind(diag(s+m))
f.con = rbind(con1,con2,con3,con4,con5)
fcon= rbind(as.matrix(f.con), con6)
```

```
#solving model
resultts = lp ("max",f.obj, fcon, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weightss1 = resultts$solution
objv= resultts$objval
} else {
weightss1 = rbind(weightss1, resultts$solution)
objv= rbind(objv, resultts$objval)
}
}
ffs = data.frame(objv, weightss1)
colnames(ffs) = c("weights"); ffs
WriteXLS(fs,"4.15-CCR-MUL-INP.xls", row.names = FALSE,col.names =
FALSE)
```

Example 5.3 Consider the data given in Table 5.1. Fuzzy efficiencies obtained from the model (5.16) for different α values are listed in Table 5.4. In this example, the nondominated sets with different α values are $S_0 = S_{0.25} = \{B, C, D, E\}$, $S_{0.5} = S_{0.75} = \{B, D\}$ and $S_1 = \{B, D\}$.

5.3.2 Leon et al.'s Approach

Leon et al. [18] extended two fuzzy versions of the classical DEA models, in particular fuzzy input-oriented BCC envelopment model (5.3), by using two ranking methods based on the comparison alpha cuts.

Let $\tilde{x}_{ij} = \left(x_{ij,1}, x_{ij,2}, x_{ij,3}, x_{ij,4}\right)$ and $\tilde{y}_{rj} = \left(y_{rj,1}, y_{rj,2}, y_{rj,3}, y_{rj,4}\right)$ be trapezoidal fuzzy input data and output data, respectively. In this case, model (5.3) is reformulated as follows regarding fuzzy arithmetic operations:

Table 5.4 The fuzzy efficiency based on model (5.25)

α	$\tilde{E}_A^*$	$\tilde{E}_B^*$	$\tilde{E}_C^*$	$\tilde{E}_D^*$	$\tilde{E}_E^*$
0.0	(0.66, 0.81, 0.99)	(0.88, 0.98, 1.09)	(0.60, 0.82, 1.12)	(0.71, 0.93, 1.25)	(0.61, 0.79, 1.02)
0.25	(0.71, 0.83, 0.96)	(0.90, 0.97, 1.04)	(0.69, 0.85, 1.04)	(0.77, 0.96, 1.20)	(0.81, 0.97, 1.15)
0.5	(0.75, 0.83, 0.92)	(0.94, 0.97, 1.00)	(0.71, 0.83, 0.97)	(0.85, 0.97, 1.12)	(0.72, 0.82, 0.93)
0.75	(0.80, 0.84, 0.88)	(0.96, 0.99, 1.02)	(0.77, 0.83, 0.90)	(0.92, 0.98, 1.05)	(0.78, 0.83, 0.89)
1.0	(0.85, 0.85, 0.85)	(1.00, 1.00, 1.00)	(0.86, 0.86, 0.86)	(1.00, 1.00, 1.00)	(1.00, 1.00, 1.00)

$$
\begin{aligned}
\min \quad & \theta_p \\
\text{s.t.} \quad & \left(\sum_{j=1}^{n}\lambda_j x_{ij,1}, \sum_{j=1}^{n}\lambda_j x_{ij,2}, \sum_{j=1}^{n}\lambda_j x_{ij,3}, \sum_{j=1}^{n}\lambda_j x_{ij,4}\right) \\
& \quad \precsim \left(\theta_p x_{ip,1}, \theta_p x_{ip,2}, \theta_p x_{ip,3}, \theta_p x_{ip,4}\right), \quad \forall i \\
& \left(\sum_{j=1}^{n}\lambda_j y_{rj,1}, \sum_{j=1}^{n}\lambda_j y_{rj,2}, \sum_{j=1}^{n}\lambda_j y_{rj,3}, \sum_{j=1}^{n}\lambda_j y_{rj,4}\right) \\
& \quad \succsim \left(y_{rp,1}, y_{rp,2}, y_{rp,3}, y_{rp,4}\right), \ \forall r \\
& \sum_{j=1}^{n}\lambda_j = 1, \\
& \lambda_j \geq 0, \quad j = 1, 2, \ldots, n.
\end{aligned}
\tag{5.26}
$$

Leon et al. [18] first used the following ranking method proposed by Ramik and Rimanek [25] to interpret the fuzzy inequalities of model (5.16).

Definition 5.5 **[18]** Let $\tilde{A} = (a_1, a_2, a_3, a_4)$ and $\tilde{B} = (b_1, b_2, b_3, b_4)$ be two trapezoidal fuzzy numbers. Then, $\tilde{A} \preceq \tilde{B}$ if and only if $a_1 \leq b_1$, $a_2 \leq b_2$, $a_3 \leq b_3$ and $a_4 \leq b_4$.

Regarding fuzzy ranking method given in Definition 5.5, model (5.26) can be reformulated as the following linear programming problem:

$$
\begin{aligned}
\min \quad & \theta_p \\
\text{s.t.} \quad & \sum_{j=1}^{n}\lambda_j x_{ij,1} \leq \theta_p x_{ip,1}, \sum_{j=1}^{n}\lambda_j x_{ij,2} \leq \theta_p x_{ip,2}, && \forall i, \\
& \sum_{j=1}^{n}\lambda_j x_{ij,3} \leq \theta_p x_{ip,3} \sum_{j=1}^{n}\lambda_j x_{ij,4} \leq \theta_p x_{ip,4}, && \forall i, \\
& \sum_{j=1}^{n}\lambda_j y_{rj,1} \geq y_{rp,1}, \sum_{j=1}^{n}\lambda_j y_{rj,2} \geq y_{rp,2}, && \forall r, \\
& \sum_{j=1}^{n}\lambda_j y_{rj,3} \geq y_{rp,3}, \sum_{j=1}^{n}\lambda_j y_{rj,4} \geq y_{rp,4}, && \forall r, \\
& \sum_{j=1}^{n}\lambda_j = 1, \\
& \lambda_j \geq 0, && j = 1, 2, \ldots, n.
\end{aligned}
\tag{5.27}
$$

In matrix form, the model (5.27) can be reformulated as follows:

$$\min \quad [1,0,\ldots,0]\left[\theta,\lambda_1,\ldots,\lambda_n\right]^{\mathrm{t}}$$

$$\text{s.t.} \quad \left[\begin{array}{c|ccc} -x_{1p,1} & x_{11,1} & \cdots & x_{1n,1} \\ \vdots & \vdots & & \vdots \\ -x_{mp,1} & x_{m1,1} & \cdots & x_{mn,1} \\ \hdashline -x_{1p,2} & x_{11,2} & \cdots & x_{1n,2} \\ \vdots & \vdots & & \vdots \\ -x_{mp,2} & x_{m1,2} & \cdots & x_{mn,2} \\ -x_{1p,3} & x_{11,3} & \cdots & x_{1n,3} \\ \vdots & \vdots & & \vdots \\ -x_{mp,3} & x_{m1,3} & \cdots & x_{mn,3} \\ \hdashline 0 & y_{11,1} & \cdots & y_{1n,1} \\ \vdots & \vdots & & \vdots \\ 0 & y_{s1,1} & \cdots & y_{sn,1} \\ 0 & y_{11,2} & \cdots & y_{1n,2} \\ \vdots & \vdots & & \vdots \\ 0 & y_{s1,2} & \cdots & y_{sn,2} \\ \hdashline 0 & y_{11.3} & \cdots & y_{1n,3} \\ \vdots & \vdots & & \vdots \\ 0 & y_{s1,3} & \cdots & y_{sn,3} \\ 0 & 1 & \cdots & 1 \end{array}\right] \left[\begin{array}{c} \theta \\ \hdashline \lambda_1 \\ \vdots \\ \lambda_n \end{array}\right] \begin{array}{c} \leq_{3m} \\ \geq_{3s} \end{array} \left[\begin{array}{c} 0 \\ y_{1p,1} \\ \vdots \\ y_{sp,1} \\ y_{1p,2} \\ \vdots \\ y_{sp,2} \\ y_{1p,3} \\ \vdots \\ y_{sp,3} \\ 1 \end{array}\right] \tag{5.28}$$

The R code for model (5.28) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data7.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df2 = data.frame(read_excel(path = "data8.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df3 = data.frame(read_excel(path = "data9.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df4 = data.frame(read_excel(path = "data10.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
```

```
#defining matrixes to be filled with data
f.conx1=matrix(ncol=N+1,nrow=m)
f.conx2=matrix(ncol=N+1,nrow=m)
f.conx3=matrix(ncol=N+1,nrow=m)
f.conx4=matrix(ncol=N+1,nrow=m)
f.cony1=matrix(ncol=N+1,nrow=s)
f.cony2=matrix(ncol=N+1,nrow=s)
f.cony3=matrix(ncol=N+1,nrow=s)
f.cony4=matrix(ncol=N+1,nrow=s)
f.conc=matrix(ncol=N+1,nrow=32)
#number of elements
m=4
s=4
N = nrow(df);N
for (j in 1:N) {
#defining right hand side, directions, and objective
f.rhs = c(rep(0,4*m),as.numeric(df1[j,(m+1):(m+s)]),as.numeric(df2[j,(m
+1):(m+s)]),
as.numeric(df3[j,(m+1):(m+s)]),as.numeric(df4[j,(m+1):(m+s)]),1)
f.dir = c(rep("<=",4*m),rep(">=",4*s),"=")
f.obj = c(1, rep(0,N))
#filling matrix of constraints coefficients
for(i in 1:m) {
f.conx1[i,1:(N+1)] = c(as.numeric(-df1[j,i]),df1[,i]) }
for(i in 1:m) {
f.conx2[i,1:(N+1)] = c(as.numeric(-df2[j,i]),df2[,i]) }
for(i in 1:m) {
f.conx3[i,1:(N+1)] = c(as.numeric(-df3[j,i]),df3[,i]) }
for(i in 1:m) {
f.conx4[i,1:(N+1)] = c(as.numeric(-df4[j,i]),df4[,i]) }
for(r in (m+1):(m+s)) {
f.cony1[(r-4),1:(N+1)] = c(0,as.numeric(df1[,r])) }
for(r in (m+1):(m+s)) {
f.cony2[(r-4),1:(N+1)] = c(0,as.numeric(df2[,r])) }
for(r in (m+1):(m+s)) {
f.cony3[(r-4),1:(N+1)] = c(0,as.numeric(df3[,r])) }
for(r in (m+1):(m+s)) {
f.cony4[(r-4),1:(N+1)] = c(0,as.numeric(df4[,r])) }
f.conx=rbind(f.conx1, f.conx2, f.conx3, f.conx4)
f.cony=rbind(f.cony1, f.cony2, f.cony3, f.cony4)
f.conc=rbind(f.conx, f.cony)
con = c(0, rep(1,N))
f.con= rbind (f.conc, con)
```

```
#solving model
results = lp('min', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weights = results$solution[1]
lambdas = results$solution[seq(2,N+1)]
} else {
weights = rbind(weights, results$solution[1])
lambdas = rbind(lambdas, results$solution[seq(2,N+1)]) } }
Es = data.frame(weights, lambdas); Es
colnames(Es) = c('effi', rep("lambda",N))
WriteXLS(Es,"4.17.xls", row.names = TRUE, col.names = TRUE)
```

The second model of Leon et al. [18] used the following fuzzy ranking method proposed by Tanaka et al. [26] which allows to decision makers to have available the efficiency results with respect to a given possibility level, or to know how the efficiency changes for different possibility levels.

Definition 5.6 [18] Let $\tilde{A} = (a_1, a_2, a_3, a_4)$ and $\tilde{B} = (b_1, b_2, b_3, b_4)$ be two trapezoidal fuzzy numbers and $\alpha \in [0, 1]$ w real number. Then, $\tilde{A} \preceq^{\alpha} \tilde{B}$ at a given possibility level α if and only if for all $h \in [\alpha, 1]$, $(1-h)a_1 + ha_2 \leq (1-h)b_1 + hb_2$ and $ha_3 + (1-h)a_4 \leq hb_3 + (1-h)b_4$.

Regarding fuzzy ranking method given in Definition 5.6, model (5.26) can be reformulated as the following linear programming problem:

$$
\begin{aligned}
\min \quad & \theta_p^{\alpha} \\
\text{s.t.} \quad & \sum_{j=1}^{n} \lambda_j \left[(1-\alpha)x_{ij,1} + \alpha x_{ij,2}\right] \leq \theta_p \left[(1-\alpha)x_{ip,1} + \alpha x_{ip,2}\right], && \forall i \\
& \sum_{j=1}^{n} \lambda_j x_{ij,2} \leq \theta_p x_{ip,2}, \quad \sum_{j=1}^{n} \lambda_j x_{ij,3} \leq \theta_p x_{ip,3}, && \forall i \\
& \sum_{j=1}^{n} \lambda_j \left[\alpha x_{ij,3} + (1-\alpha)x_{ij,4}\right] \leq \theta_p \left[\alpha x_{ip,3} + (1-\alpha)x_{ip,4}\right], && \forall i \\
& \sum_{j=1}^{n} \lambda_j \left[(1-\alpha)y_{rj,1} + \alpha y_{rj,2}\right] \geq \left[(1-\alpha)y_{rp,1} + \alpha y_{rp,2}\right], && \forall r \\
& \sum_{j=1}^{n} \lambda_j y_{rj,2} \geq y_{rp,2}, \quad \sum_{j=1}^{n} \lambda_j y_{rj,3} \geq y_{rp,3}, && \forall r \\
& \sum_{j=1}^{n} \lambda_j \left[\alpha y_{rj,3} + (1-\alpha)y_{rj,4}\right] \geq \left[\alpha y_{rp,3} + (1-\alpha)y_{rp,4}\right], && \forall r \\
& \sum_{j=1}^{n} \lambda_j = 1, \\
& \lambda_j \geq 0, && j = 1, 2, \ldots, n.
\end{aligned}
\tag{5.29}
$$

The efficiency score of DMU_p at the α possibility level is given by θ_p^{α}. By solving model (5.29) for different values of α the analyst can observe how the efficiency scores of the DMUs change when the possibility level α varies.

Assume

$$\begin{aligned}
\dot{x}_{ij,1} &= \left[(1-\alpha)x_{ij,1} + \alpha x_{ij,2}\right] && \forall i, \forall j, \\
\dot{x}_{ip,1} &= \left[(1-\alpha)x_{ip,1} + \alpha x_{ip,2}\right] && \forall i, j = p, \\
\ddot{x}_{ij,4} &= \left[\alpha x_{ij,3} + (1-\alpha)x_{ij,4}\right] && \forall i, \forall j, \\
\ddot{x}_{ip,4} &= \left[\alpha x_{ip,3} + (1-\alpha)x_{ip,4}\right] && \forall i, j = p, \\
\dot{y}_{rj,1} &= \left[(1-\alpha)y_{rj,1} + \alpha y_{rj,2}\right] && \forall r, \forall j \\
y_{rp,1} &= \left[(1-\alpha)y_{rp,1} + \alpha y_{rp,2}\right] && \forall r, j = p \\
\ddot{y}_{rj,4} &= \left[\alpha y_{rj,3} + (1-\alpha)y_{rj,4}\right] && \forall r, \forall j \\
\ddot{y}_{rp,4} &= \left[\alpha y_{rp,3} + (1-\alpha)y_{rp,4}\right] && \forall r, j = p
\end{aligned} \tag{5.30}$$

In matrix form, the model (5.29) can be stated as follows:

$$\begin{aligned}
\min \quad & [1,0,\ldots,0]\left[\theta, \lambda_1, \ldots, \lambda_n\right]^{\mathrm{t}} \\
\text{s.t.} \quad & \begin{bmatrix} \dot{x}_p & \dot{x}_j \\ x_{p,2} & x_{j,2} \\ x_{p,3} & x_{j,3} \\ \ddot{x}_p & \ddot{x}_j \\ 0 & \dot{y}_{rj} \\ 0 & y_{j,2} \\ 0 & y_{j,3} \\ 0 & \ddot{y}_j \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \hdashline \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix} \begin{matrix} \leq_{4m} \\ \geq_{4s} \end{matrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \dot{y}_p \\ y_{p,2} \\ y_{p,3} \\ \ddot{y}_p \end{bmatrix} \\
& \lambda \geq 0.
\end{aligned} \tag{5.31}$$

The R code for model (5.31) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data7.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
```

```
df2 = data.frame(read_excel(path = "data8.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df3 = data.frame(read_excel(path = "data9.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df4 = data.frame(read_excel(path = "data10.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
f.conx1=matrix(ncol=N+1,nrow=m)
f.conx2=matrix(ncol=N+1,nrow=m)
f.conx3=matrix(ncol=N+1,nrow=m)
f.conx4=matrix(ncol=N+1,nrow=m)
f.cony1=matrix(ncol=N+1,nrow=s)
f.cony2=matrix(ncol=N+1,nrow=s)
f.cony3=matrix(ncol=N+1,nrow=s)
f.cony4=matrix(ncol=N+1,nrow=s)
f.conc=matrix(ncol=N+1,nrow=32)
#parameter and number od elements
alpha=0.05
m=4
s=ncol(df1)-m
N = nrow(df1)
for (j in 1:N) {
#defining right hand side, directions, and obective
f.rhs = c(rep(0,4*m),(1-alpha)*as.numeric(df1[j, (m+1):(m+s)])+alpha*as.
numeric(df2[j,(m+1):(m+s)]),as.numeric(df2[j,(m+1):(m+s)]),
as.numeric(df3[j,(m+1):(m+s)]),alpha*as.numeric(df3[j,(m+1):(m+s)])
+(1-alpha)*as.numeric(df4[j,(m+1):(m+s)]),1)
f.dir = c(rep("<=",4*m),rep(">=",4*s),"=")
f.obj = c(1, rep(0,N))
#filling matrix of constratints'coefficient
for(i in 1:m) {
f.conx1[i,1:(N+1)] = c(-((1-alpha)*as.numeric(df1[j,i])+alpha*as.numeric
(df2[j,i])),((1-alpha)*df1[,i]+alpha*df2[,i])) }
for(i in 1:m) {
f.conx2[i,1:(N+1)] = c(as.numeric(-df2[j,i]),df2[,i]) }
for(i in 1:m) {
f.conx3[i,1:(N+1)] = c(as.numeric(-df3[j,i]),df3[,i]) }
for(i in 1:m) {
f.conx4[i,1:(N+1)] = c(-(alpha*as.numeric(df3[j,i])+(1-alpha)*df4[j,i]),
alpha*df3[,i]+(1-alpha)*df4[,i]) }
for(r in (m+1):(m+s)) {
```

```
f.cony1[(r-4),1:(N+1)] = c(0,(1-alpha)*as.numeric(df1[,r])+alpha*as.nu-
meric(df2[,r])) }
for(r in (m+1):(m+s)) {
f.cony2[(r-4),1:(N+1)] = c(0,as.numeric(df2[,r])) }
for(r in (m+1):(m+s)) {
f.cony3[(r-4),1:(N+1)] = c(0,as.numeric(df3[,r])) }
for(r in (m+1):(m+s)) {
f.cony4[(r-4),1:(N+1)] = c(0,alpha*as.numeric(df3[,r])+(1-alpha)*as.numeric
(df4[,r])) }
f.conx=rbind(f.conx1, f.conx2, f.conx3, f.co nx4)
f.cony=rbind(f.cony1, f.cony2, f.cony3, f.cony4)
f.conc=rbind(f.conx, f.cony)
con = c(0, rep(1,N))
f.con= rbind (f.conc, con)
#solving model
resultss = lp('min', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weightss = resultss$solution[1]
lambdass = resultss$solution[seq(2,N+1)]
} else {
weightss = rbind(weightss, resultss$solution[1])
lambdass = rbind(lambdass, resultss$solution[seq(2,N+1)]) } }
EEs = data.frame(weightss)
colnames(Es) = c('effi', rep("lambda",N))
WriteXLS(Es,"4.18.xls", row.names = TRUE, col.names = TRUE)
```

Example 5.4 Consider the data given in Table 5.1. Results of evaluating the efficiency of data with models (5.27) and (5.29) are recorded in Table 5.5. The efficiency scores obtained from model (5.29) correspond to the $h = 0$ row. We can see that DMUs B, D, and E are crisp efficiecnt.

Table 5.5 The fuzzy efficiency based on model (5.18)

α	$\tilde{E}_A^*$	$\tilde{E}_B^*$	$\tilde{E}_C^*$	$\tilde{E}_D^*$	$\tilde{E}_E^*$
0.0	0.94	1.00	1.00	1.00	1.00
0.25	0.93	1.00	1.00	1.00	1.00
0.5	0.91	1.00	1.00	1.00	1.00
0.75	0.90	1.00	0.97	1.00	1.00
1.0	0.89	1.00	0.93	1.00	1.00

5.3.3 Soleimani-damaneh et al.'s Approach

Solemani-damaneh et al. [19] have used one ranking method belong to the class of distance-based approach for providing a technique for treating the fuzzy data in DEA framework. They have used the following singed distance proposed by Yao and Wu [27] to interpret the fuzzy inequalities of model (5.26).

Definition 5.7 [27] Let $\tilde{A}$ and $\tilde{B}$ be two fuzzy numbers. The signed distance is defined as follows:

$$d\left(\tilde{A},\tilde{B}\right)=\frac{1}{2}\int_0^1\left(\left(\tilde{A}\right)_\alpha^L+\left(\tilde{A}\right)_\alpha^U-\left(\tilde{B}\right)_\alpha^L-\left(\tilde{B}\right)_\alpha^U\right)d\alpha \tag{5.32}$$

Then, $\tilde{A}\preceq\tilde{B}$ if and inly if $d\left(\tilde{A},\tilde{B}\right)\leq 0$.

Remark 5.3 For two trapezoidal fuzzy numbers $\tilde{A}=(a_1,a_2,a_3,a_4)$ and $\tilde{B}=(b_1,b_2,b_3,b_4)$, Definition 5.6 is reduced to

$$\tilde{A}\lesssim\tilde{B}\Leftrightarrow a_1+a_2+a_3+a_4\leq b_1+b_2+b_3,\ +b_4$$

Regarding Remark 5.3, the fuzzy input-oriented BCC envelopment model can be reformulated as follows:

$$\begin{aligned}
\min\quad & \theta_p\\
\text{s.t.}\quad & \sum_{j=1}^{n}\lambda_j\left(x_{ij,1}+x_{ij,2}+x_{ij,3}+x_{ij,4}\right)\leq\theta_p\left(x_{ip,1}+x_{ip,2}+x_{ip,3}+x_{ip,4}\right), && \forall i\\
& \sum_{j=1}^{n}\lambda_j\left(y_{rj,1}+y_{rj,2}+y_{rj,3}+y_{rj,4}\right)\geq\left(y_{rp,1}+y_{rp,2}+y_{rp,3}+y_{rp,4}\right), && \forall r\\
& \sum_{j=1}^{n}\lambda_j=1,\\
& \lambda_j\geq 0, && j=1,2,\ldots,n.
\end{aligned} \tag{5.33}$$

Assume

$$\begin{aligned}
\dot{x}_{ij}&=\left(x_{ij,1}+x_{ij,2}+x_{ij,3}+x_{ij,4}\right) && \forall i,\forall j\\
\dot{x}_{ip}&=\left(x_{ip,1}+x_{ip,2}+x_{ip,3}+x_{ip,4}\right) && \forall i,j=p\\
\dot{y}_{rj}&=\left(y_{rj,1}+y_{rj,2}+y_{rj,3}+y_{rj,4}\right) && \forall r,\forall j\\
\dot{y}_{rp}&=\left(y_{rp,1}+y_{rp,2}+y_{rp,3}+y_{rp,4}\right) && \forall r,j=p
\end{aligned} \tag{5.34}$$

In matrix form, model (5.34) can be stated as follows:

$$\begin{aligned}
\min \quad & [1,0,...,0]\left[\theta,\lambda_1,...,\lambda_n\right]^{\mathrm{t}} \\
\text{s.t.} \quad & \left[\begin{array}{c|ccc} -\dot{x}_{1p} & \dot{x}_{11} & \cdots & \dot{x}_{1n} \\ \vdots & \vdots & & \vdots \\ -\dot{x}_{mp} & \dot{x}_{m1} & \cdots & \dot{x}_{mn} \\ \hline 0 & \dot{y}_{11} & \cdots & \dot{y}_{1n} \\ \vdots & \vdots & & \vdots \\ 0 & \dot{y}_{s1} & \cdots & \dot{y}_{sn} \\ 0 & 1 & \cdots & 1 \end{array}\right] \left[\begin{array}{c} \theta \\ \hline \lambda_1 \\ \vdots \\ \lambda_n \end{array}\right] \begin{array}{c} \leq_m \\ \hline \geq_s \\ =_1 \end{array} \left[\begin{array}{c} 0_{m\times 1} \\ \hline y_{1p} \\ \vdots \\ y_{sp} \\ \hline -- \\ 1 \end{array}\right]
\end{aligned} \tag{5.35}$$

The R code for model (5.35) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#required libraries
df1 = data.frame(read_excel(path = "data7.xlsx", sheet="1"))
df2 = data.frame(read_excel(path = "data8.xlsx", sheet="1"))
df3 = data.frame(read_excel(path = "data9.xlsx", sheet="1"))
df4 = data.frame(read_excel(path = "data10.xlsx", sheet="1"))
#defining a matrix to be filled with data
f.con=matrix(ncol=N+1,nrow=m+s)
#number of elements
m=4
s = ncol(df1)-m;s
N = nrow(df1);N
for (j in 1:N) {
#defining right hand side, directions, and objective
f.rhs = c(rep(0,m),(as.numeric(df1[j,(m+1):(m+s)])+as.numeric(df2[j,(m+1):
(m+s)])+as.numeric(df3[j,(m+1):(m+s)])+as.numeric(df4[j,(m+1):(m+s)])))
f.dir = c(rep("<=",m),rep(">=",s))
f.obj = c(1, rep(0,N))
#filling matrix of constraints' coefficient
for(i in 1:m){
f.con[i,1:(N+1)] = c(as.numeric(-df1[j,i])+as.numeric(-df2[j,i])+as.numeric
(-df3[j,i])+as.numeric(-df4[j,i]),(df1[,i]+df2[,i]+df3[,i]+df4[,i]))
}
for(r in (m+1):(s+m)) {
```

```
f.con[r,1:(N+1)] = c(0,as.numeric(df1[,r])+as.numeric(df2[,r])+as.numeric
(df3[,r])+as.numeric(df4[,r]))
}
#solving model
results = lp('min', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weights = results$solution[1]
lambdas = results$solution[seq(2,N+1)]
} else {
weights = rbind(weights, results$solution[1])
lambdas = rbind(lambdas, results$solution[seq(2,N+1)]) } }
Es = data.frame(weights, lambdas); Es
colnames(Es) = c('effi', rep("lambda",N),rep("x benchmark", m), rep("y
benchmark", s))
WriteXLS(Es,"1-CCR-ENV_INP.xls", row.names = TRUE, col.names = TRUE)
```

Example 5.5 Consider the data given in Table 5.1. The results of evaluating the efficiency of units are listed in Table 5.6. From Table 5.5 it can be seen that DMUs B, D, and E are recognized as efficiecnt DMUs.

5.4 The Possibility Approach

In this section, a possibility approach as a way to deal with the uncertainty in fuzzy objectives and fuzzy constraints of fuzzy DEA models through the use of possibility measures is presented [28–30].

Lertworasirikul et al. [28] have proposed "possibility approach" and "credibility approach" for solving the ranking problem in FDEA models. According to the possibility approach, the FDEA model is transformed into possibility DEA model by using possibility measures of fuzzy events (fuzzy constraints). Moreover, according to the credibility approach, an efficiency value for each DMU is obtained as a representative of its possible range.

Lertworasirikul et al. [28] have used the concept of chance-constrained programming (CCP) [31], to solve fuzzy input-oriented CCR multiplier model (5.2). Using the concepts of CCP and possibility of fuzzy events, the model (5.2) becomes the following possibility DEA model:

Table 5.6 The efficiency based on model (5.33)

DMU	*A*	*B*	*C*	*D*	*E*
Efficiency	0.89	1.00	0.94	1.00	1.00

$$
\begin{aligned}
\max \quad & \bar{f} \\
\text{s.t.} \quad & \pi\left(\sum_{r=1}^{s} u_r \tilde{y}_{rp} \geq \bar{f}\right) \geq \beta \\
& \pi\left(\sum_{i=1}^{m} v_i \tilde{x}_{ip} = 1\right) \geq \beta_0, \\
& \pi\left(\sum_{r=1}^{s} u_r \tilde{y}_{rj} - \sum_{i=1}^{m} v_i \tilde{x}_{ij} \leq 0\right) \geq \beta_j, \quad j = 1, 2, \ldots, n, \\
& v_i \geq 0, \qquad\qquad i = 1, 2, \ldots, m, \\
& u_r \geq 0, \qquad\qquad r = 1, 2, \ldots, s.
\end{aligned}
\tag{5.36}
$$

where $\beta, \beta_0, \beta_1, \ldots, \beta_n \in [0, 1]$ are pre-specified acceptable levels of possibility for the objective and the constraints, respectively, and $\pi(\tilde{A})$ denotes the possibility of the fuzzy event $\tilde{A}$ that is given by Defintion 1.23. To make a reasonable efficiency comparison of DMUs, all fuzzy constraints should be satisfied with same possibility level, i.e., $\beta, = \beta_0, = \beta_1 = \cdots = \beta_n = \alpha$.

Remark 5.4 For a trapezoidal fuzzy number $\tilde{A} = (a_1, a_2, a_3, a_4)$ we have

- $\pi(\tilde{A} \geq a) \geq \alpha \Leftrightarrow (\tilde{A})_{\alpha}^{U} = a_4 - (a_4 - a_3)\alpha \geq a$,
- $\pi(\tilde{A} = a) \geq \alpha \Leftrightarrow (\tilde{A})_{\alpha}^{L} = a_1 + (a_2 - a_1)\alpha \leq a$, $(\tilde{A})_{\alpha}^{U} = a_4 - (a_4 - a_3)\alpha$,
- $\pi(\tilde{A} \leq a) \geq \alpha \Leftrightarrow (\tilde{A})_{\alpha}^{L} = a_1 + (a_2 - a_1)\alpha \leq a$.

When the input and output data are assumed to be trapezoidal fuzzy numbers, regarding Remark 5.4, model (5.36) can be reformulated as the following linear programming problem:

$$
\begin{aligned}
\max \quad & \bar{f} \\
\text{s.t.} \quad & \sum_{r=1}^{s} u_r \left[y_{rp,4} - (y_{rp,4} - y_{rp,3})\alpha\right] \geq \bar{f} \\
& \sum_{i=1}^{m} v_i \left[x_{ip,1} + (x_{ip,2} - x_{ip,1})\alpha\right] \leq 1, \\
& \sum_{i=1}^{m} v_i \left[x_{ip,4} + (x_{ip,4} - x_{ip,3})\alpha\right] \geq 1, \\
& \sum_{r=1}^{s} u_r \left[y_{rj,1} - (y_{rj,2} - y_{rj,1})\alpha\right] \leq \sum_{i=1}^{m} v_i \left[x_{ij,4} + (x_{ij,4} - x_{ij,3})\alpha\right], \quad j = 1, 2, \ldots, n, \\
& v_i \geq 0, \qquad\qquad i = 1, 2, \ldots, m, \\
& u_r \geq 0, \qquad\qquad r = 1, 2, \ldots, s.
\end{aligned}
\tag{5.37}
$$

Definition 5.8 A DMU is α-possibilistic efficient if $\bar{f}$ at the α-possibility level is greater than or equal to one; otherwise, it is α-possibilistic inefficient.

Assume

$$\begin{aligned}
\dot{y}_{rp} &= \left[y_{rp,4} - (y_{rp,4} - y_{rp,3})\alpha\right] \quad \forall r, j = p \\
\dot{x}_{ip} &= \left[x_{ip,1} + (x_{ip,2} - x_{ip,1})\alpha\right] \quad \forall i, j = p \\
\ddot{x}_{ip} &= \left[x_{ip,4} + (x_{ip,4} - x_{ip,3})\alpha\right] \quad \forall i, j = p \\
\dot{y}_{rj} &= \left[y_{rj,1} - (y_{rj,2} - y_{rj,1})\alpha\right] \quad \forall r, \forall j \\
\ddot{x}_{ij} &= \left[x_{ij,4} + (x_{ij,4} - x_{ij,3})\alpha\right] \quad \forall i, \forall j
\end{aligned} \tag{5.38}$$

In matrix form, the model (5.37) can be stated as follows:

$$\max \quad [0,...,0,0,...,0,1]\left[v_1,...,v_m,u_1,...,u_s,\overline{f}\right]^{\mathrm{t}}$$

$$\text{s.t.} \quad \left[\begin{array}{ccc:ccc:c}
-\ddot{x}_{11} & \cdots & -\ddot{x}_{m1} & \dot{y}_{11} & \cdots & \dot{y}_{s1} & 0 \\
\vdots & & \vdots & \vdots & & \vdots & \vdots \\
-\ddot{x}_{1n} & \cdots & -\ddot{x}_{mn} & \dot{y}_{1n} & \cdots & \dot{y}_{sn} & 0 \\
\hdashline
\dot{x}_{1p} & \cdots & \dot{x}_{mp} & 0 & \cdots & 0 & 0 \\
\ddot{x}_{1p} & \cdots & \ddot{x}_{mp} & 0 & & 0 & 0 \\
0 & \cdots & 0 & \dot{y}_{1p} & \cdots & \dot{y}_{sp} & -1
\end{array}\right]
\left[\begin{array}{c}
v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \\ \overline{f}
\end{array}\right]
\begin{array}{c}\geq \\ \leq \\ \hdashline \geq \\ \leq \end{array}
\left[\begin{array}{c}0\\1\\ \hdashline 1\\0\end{array}\right] \tag{5.39}$$

$$v_i \geq 0, \quad i = 1,2,...,m,$$
$$u_r \geq 0, \quad r = 1,2,...,s.$$

The R code for model (5.39) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data7.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE);df1
df2 = data.frame(read_excel(path = "data8.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE);df2
df3 = data.frame(read_excel(path = "data9.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df4 = data.frame(read_excel(path = "data10.xlsx", sheet="1"), check.-
names = FALSE, fix.empty.names = TRUE)
#parameter, and number of elements
m=4
s = ncol(df1)-m;s
```

```
N = nrow(df1);N
alpha=1
#defining right hand side, directions, and objective
f.rhs = c(0,1,1, rep(0,N))
f.dir = c(">=","<=",">=",rep("<=",N))
f.obj = c(1, rep(0,m), rep(0,s))
for (j in 1:N) {
#defining matrix of constraints' coefficint
con1= c(-1, rep(0,m), (as.numeric(df4[j,(m+1):(m+s)])-(as.numeric(df4[j,(m
+1):(m+s)])-as.numeric(df3[j,(m+1):(m+s)]))*alpha))
con2= c(0, (as.numeric(df1[j,1:m])+(as.numeric(df2[j,1:m])-as.numeric(df1
[j,1:m]))*alpha), rep(0,s))
con3= c(0, (as.numeric(df4[j,1:m])+(as.numeric(df4[j,1:m])-as.numeric(df3
[j,1:m]))*alpha), rep(0,s))
con4= cbind(0, -(df4[,1:m]+(df4[,1:m]-df3[,1:m])*alpha), (df1[,(m+1):(m
+s)]-(df2[,(m+1):(m+s)]-df1[,(m+1):(m+s)])*alpha))
f.con1 = rbind(con1, (con2), (con3), (con4))
#solving model
result= lp('max', f.obj, f.con1, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weightss = result$solution
effvrss = result$objval
} else {
weightss = rbind(weightss,result$solution)
effvrss = rbind(effvrss, result$objval) } }
ECs = data.frame(weightss, effvrss); ECs
colnames(ECs) = c('solution', "effi")
WriteXLS(ECs,"model.xls", row.names = TRUE, col.names = TRUE)
```

Example 5.6 Consider the data given in Table 5.1. The results for five different possibility levels based on mdeol (5.37) are provided in Table 5.7. The results indicate DMUs B, D and E are possibilistically efficient at all possibility levels, whereas DMUs A and E are possibilistically efficient only at some possibility levels.

Table 5.7 The fuzzy efficiency based on model (5.37)

α	$\tilde{E}_A^*$	$\tilde{E}_B^*$	$\tilde{E}_C^*$	$\tilde{E}_D^*$	$\tilde{E}_E^*$
0.0	1.107	1.238	1.276	1.52	1.296
0.25	1.032	1.173	1.149	1.386	1.226
0.5	0.963	1.112	1.035	1.258	1.159
0.75	0.904	1.055	0.932	1.131	1.095
1.0	0.855	1.000	0.861	1.000	1.000

The credibility approach uses the "expected credits" of fuzzy variables to deal with the uncertainty in fuzzy objectives and fuzzy constraints [29]. The approach transforms fuzzy input-oriented BCC multiplier model (5.4) into following credibility programming-DEA model:

$$\begin{aligned}
\max \quad & E\left(\sum_{r=1}^{s} u_r \tilde{y}_{rp} + u_0\right) \\
\text{s.t.} \quad & E\left(\sum_{i=1}^{m} v_i \tilde{x}_{ip}\right) = 1, \\
& E\left(\sum_{r=1}^{s} u_r \tilde{y}_{rj} - \sum_{i=1}^{m} v_i \tilde{x}_{ij} + u_0\right) \le 0, \quad j = 1, 2, \ldots, n, \\
& v_i \ge 0, \qquad\qquad i = 1, 2, \ldots, m, \\
& u_r \ge 0, \qquad\qquad r = 1, 2, \ldots, s.
\end{aligned} \tag{5.40}$$

where $E(\tilde{A})$ denotes the expected credit of the fuzzy event $\tilde{A}$ (see Definition 1.24).

Remark 5.5 The expected credit of a trapezoidal fuzzy number $\tilde{A} = (a_1, a_2, a_3, a_4)$ is given as $E(\tilde{A}) = \frac{1}{4}(a_1 + a_2 + a_3 + a_4)$.

When the input and output data are assumed to be trapezoidal fuzzy numbers, regarding Remark 5.5, model (5.23) can be reformulated as the following linear programming problem:

$$\begin{aligned}
\max \quad & \tfrac{1}{4}\left[\sum_{r=1}^{s} u_r \left[y_{rp,1} + y_{rp,2} + y_{rp,3} + y_{rp,4}\right]\right] + u_o \\
\text{s.t.} \quad & \tfrac{1}{4}\sum_{i=1}^{m} v_i \left[x_{ip,1} + x_{ip,2} + x_{ip,3} + x_{ip,4}\right] = 1, \\
& \frac{1}{4}\left[\sum_{r=1}^{s} u_r \left[y_{rj,1} + y_{rj,2} + y_{rj,3} + y_{rj,4}\right]\right. \\
& \left. - \sum_{i=1}^{m} v_i \left[x_{ij,1} + x_{ij,2} + x_{ij,3} + x_{ij,4}\right]\right] + u_o \le 0, \quad \forall j, \\
& v_i \ge 0, \qquad\qquad i = 1, 2, \ldots, m, \\
& u_r \ge 0, \qquad\qquad r = 1, 2, \ldots, s.
\end{aligned} \tag{5.41}$$

Definition 5.9 DMU_p is called a credibilistically efficient DMU if $E\left(\sum_{r=1}^{s} u_r \tilde{y}_{rp} + u_0\right) = 1$; otherwise, it is a credibilistically inefficient DMU.

Assume

$$\begin{aligned}
\dot{y}_{rj} &= \left(\tfrac{1}{4}\right)\left[y_{rj,1} + y_{rj,2} + y_{rj,3} + y_{rj,4}\right] && \forall r, \forall j \\
\dot{x}_{ij} &= \left(\tfrac{1}{4}\right)\left[x_{ij,1} + x_{ij,2} + x_{ij,3} + x_{ij,4}\right] && \forall i, \forall j \\
\dot{y}_{rp} &= \left(\tfrac{1}{4}\right)\left[y_{rp,1} + y_{rp,2} + y_{rp,3} + y_{rp,4}\right] && \forall r, j = p \\
\dot{x}_{ip} &= \left(\tfrac{1}{4}\right)\left[x_{ip,1} + x_{ip,2} + x_{ip,3} + x_{ip,4}\right] && \forall i, j = p
\end{aligned} \tag{5.42}$$

In matrix form, the model (5.41) can be stated as follows:

$$\begin{aligned} \max \quad & \left[0,...,0,y_{1p},...,y_{sp}\right]\left[v_1,...,v_m,u_1,...,u_s\right]^{\mathrm{t}} \\ \text{s.t.} \quad & \left[\begin{array}{ccc:ccc} -\dot{x}_{11} & \cdots & -\dot{x}_{m1} & \dot{y}_{11} & \cdots & \dot{y}_{s1} \\ \vdots & & \vdots & \vdots & & \vdots \\ -\dot{x}_{1n} & \cdots & -\dot{x}_{mn} & \dot{y}_{1n} & \cdots & \dot{y}_{sn} \\ \hdashline \dot{x}_{1p} & \cdots & \dot{x}_{mp} & 0 & \cdots & 0 \end{array}\right] \left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \end{array}\right] \begin{array}{l} \leq_n \\ \hdashline =_1 \end{array} \left[\begin{array}{c} 0_{n\times 1} \\ \hdashline 1 \end{array}\right] \end{aligned} \tag{5.43}$$

The R code for model (5.43) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data7.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df2 = data.frame(read_excel(path = "data8.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df3 = data.frame(read_excel(path = "data9.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df4 = data.frame(read_excel(path = "data10.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
#number of elements
m=4
s = ncol(df1)-m
N = nrow(df1)
#defining right hand side and directions
f.rhs = c(1, rep(0,N))
f.dir = c("=",rep("<=",N))
for (j in 1:N) {
#defining objective and matrix of constraints' coefficient
f.obj = c((1/4)*(as.numeric(df1[j,(m+1):(m+s)])+as.numeric(df2[j,(m+1):(m
+s)])+as.numeric(df3[j,(m+1):(m+s)])+as.numeric(df4[j,(m+1):(m+s)])),
rep(0,m), 1, -1)
con1= c(rep(0,s), (1/4)*(as.numeric(df1[j,1:m])+as.numeric(df2[j,1:m])+as.
numeric(df3[j,1:m])+as.numeric(df4[j,1:m])), 0, 0)
```

```
con2= cbind((1/4)*(df1[,(m+1):(m+s)]+df2[,(m+1):(m+s)]+df3[,(m+1):(m+s)]
+df4[,(m+1):(m+s)]), (-1/4)*((df1[,1:m])+(df2[,1:m])+(df3[,1:m])+(df4[,1:m])),
1, -1)
f.con = rbind(con1, con2)
#solving model
result = lp('max', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weight = result$solution
effvr = result$objval
} else {
weight = rbind(weight,result$solution)
effvr = rbind(effvr, result$objval) } }
EEs = data.frame(weight, effvr)
colnames(Es) = c('solution', "effi")
WriteXLS(Es, "n.models.xls", row.names = TRUE, col.names = TRUE)
```

Example 5.7 Consider the data given in Table 5.1. The efficiency values from the model (5.41) obtained by credibility approach are given in Table 5.8. The results indicate DMUs B, D and E are credibilistically efficient, while DMUs A and C are credibilistically inefficient according to Definition 5.9.

5.5 The Fuzzy Arithmetic Approach

In this section, three approaches based on fuzzy arithmetic are presented to deal with fuzziness in input and output data in DEA [32–34].

5.5.1 Wang et al.'s Approach

The main idea behind of Wang et al.'s approach [32] is to convert the fractional form of each FDEA model into several crisp fractional DEA model from the perspective of fuzzy arithmetic. The idea comes from this fact that a fuzzy fractional DEA model cannot be transformed into a linear DEA model in the traditional way that we do for a crisp fractional DEA model.

Table 5.8 The efficiency based on model (5.41)

DMU	*A*	*B*	*C*	*D*	*E*
Efficiency	0.889	1.000	0.935	1.000	1.000

When the input and output data are assumed to be triangular fuzzy numbers, the fuzzy efficiency is defined as follows:

$$\tilde{\theta}_j = \frac{\sum_{r=1}^{s} u_r \tilde{y}_{rj}}{\sum_{i=1}^{m} v_i \tilde{x}_{ij}} = \frac{\left(\sum_{r=1}^{s} u_r y_{rj,1}, \sum_{r=1}^{s} u_r y_{rj,2}, \sum_{r=1}^{s} u_r y_{rj,3}\right)}{\left(\sum_{i=1}^{m} v_i x_{ij,1}, \sum_{i=1}^{m} v_i x_{ij,2}, \sum_{i=1}^{m} v_i x_{ij,3}\right)} \approx \left(\frac{\sum_{r=1}^{s} u_r y_{rj,1}}{\sum_{i=1}^{m} v_i x_{ij,3}}, \frac{\sum_{r=1}^{s} u_r y_{rj,2}}{\sum_{i=1}^{m} v_i x_{ij,2}}, \frac{\sum_{r=1}^{s} u_r y_{rj,3}}{\sum_{i=1}^{m} v_i x_{ij,1}}\right)$$

Wang et al. [32] developed two following fuzzy fractional DEA models based on $\tilde{\theta}_j \leq 1$ or $\tilde{\theta}_j \precsim \tilde{1}$ in order to measure the efficiency of each DMU relative to other DMUs:

$$\begin{aligned} \max \quad & \tilde{\theta}_p = \left(\theta_{p,1}, \theta_{p,2}, \theta_{p,3}\right) \approx \left(\frac{\sum_{r=1}^{s} u_r y_{rp,1}}{\sum_{i=1}^{m} v_i x_{ip,3}}, \frac{\sum_{r=1}^{s} u_r y_{rp,2}}{\sum_{i=1}^{m} v_i x_{ip,2}}, \frac{\sum_{r=1}^{s} u_r y_{rp,3}}{\sum_{i=1}^{m} v_i x_{ip,1}}\right) \\ \text{s.t.} \quad & \tilde{\theta}_j = \left(\theta_{j,1}, \theta_{j,2}, \theta_{j,3}\right) \approx \left(\frac{\sum_{r=1}^{s} u_r y_{rj,1}}{\sum_{i=1}^{m} v_i x_{ij,3}}, \frac{\sum_{r=1}^{s} u_r y_{rj,2}}{\sum_{i=1}^{m} v_i x_{ij,2}}, \frac{\sum_{r=1}^{s} u_r y_{rj,3}}{\sum_{i=1}^{m} v_i x_{ij,1}}\right) \leq 1, \quad \forall j, \\ & v_i \geq 0, \quad i = 1, 2, \ldots, m, \\ & u_r \geq 0, \quad r = 1, 2, \ldots, s. \end{aligned} \tag{5.44}$$

$$\begin{aligned} \max \quad & \tilde{\theta}_p = \left(\theta_{p,1}, \theta_{p,2}, \theta_{p,3}\right) \approx \left(\frac{\sum_{r=1}^{s} u_r y_{rp,1}}{\sum_{i=1}^{m} v_i x_{ip,3}}, \frac{\sum_{r=1}^{s} u_r y_{rp,2}}{\sum_{i=1}^{m} v_i x_{ip,2}}, \frac{\sum_{r=1}^{s} u_r y_{rp,3}}{\sum_{i=1}^{m} v_i x_{ip,1}}\right) \\ \text{s.t.} \quad & \tilde{\theta}_j = \left(\theta_{j,1}, \theta_{j,2}, \theta_{j,3}\right) \approx \left(\frac{\sum_{r=1}^{s} u_r y_{rj,1}}{\sum_{i=1}^{m} v_i x_{ij,3}}, \frac{\sum_{r=1}^{s} u_r y_{rj,2}}{\sum_{i=1}^{m} v_i x_{ij,2}}, \frac{\sum_{r=1}^{s} u_r y_{rj,3}}{\sum_{i=1}^{m} v_i x_{ij,1}}\right) \precsim \tilde{1}, \quad \forall j, \\ & v_i \geq 0, \quad i = 1, 2, \ldots, m, \\ & u_r \geq 0, \quad r = 1, 2, \ldots, s. \end{aligned} \tag{5.45}$$

From model (5.44), $\tilde{\theta}_j = \left(\theta_{j,1}, \theta_{j,2}, \theta_{j,3}\right) \leq 1$ is reduced to $\theta_{j,3} \leq 1$ and therefore this model is simplified ad follows:

$$\begin{aligned} \max \quad & \tilde{\theta}_p = \left(\theta_{p,1}, \theta_{p,2}, \theta_{p,3}\right) \approx \left(\frac{\sum_{r=1}^{s} u_r y_{rp,1}}{\sum_{i=1}^{m} v_i x_{ip,3}}, \frac{\sum_{r=1}^{s} u_r y_{rp,2}}{\sum_{i=1}^{m} v_i x_{ip,2}}, \frac{\sum_{r=1}^{s} u_r y_{rp,3}}{\sum_{i=1}^{m} v_i x_{ip,1}}\right) \\ \text{s.t.} \quad & \theta_{j,3} = \frac{\sum_{r=1}^{s} u_r y_{rj,3}}{\sum_{i=1}^{m} v_i x_{ij,1}} \leq 1, \quad j = 1, 2, \ldots, n, \\ & v_i \geq 0, \qquad i = 1, 2, \ldots, m, \\ & u_r \geq 0, \qquad r = 1, 2, \ldots, s. \end{aligned} \tag{5.46}$$

Wang et al.'s [32] constructed the following three crisp linear DEA models to find each component of the fuzzy efficiency for the DMU under consideration in model (5.46):

$$\begin{aligned}
\max \quad & \theta_{p,1} = \sum_{r=1}^{s} u_r y_{rp,1} \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ip,3} = 1 \\
& \sum_{r=1}^{s} u_r y_{rj,3} - \sum_{i=1}^{m} v_i x_{ij,1} \le 1, \quad j = 1,2,\ldots,n, \\
& v_i \ge 0, \qquad\qquad i = 1,2,\ldots,m, \\
& u_r \ge 0, \qquad\qquad r = 1,2,\ldots,s.
\end{aligned} \tag{5.47}$$

$$\begin{aligned}
\max \quad & \theta_{p,2} = \sum_{r=1}^{s} u_r y_{rp,2} \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ip,2} = 1, \\
& \sum_{r=1}^{s} u_r y_{rj,3} - \sum_{i=1}^{m} v_i x_{ij,1} \le 1, \quad j = 1,2,\ldots,n, \\
& v_i \ge 0, \qquad\qquad i = 1,2,\ldots,m, \\
& u_r \ge 0, \qquad\qquad r = 1,2,\ldots,s.
\end{aligned} \tag{5.48}$$

$$\begin{aligned}
\max \quad & \theta_{p,3} = \sum_{r=1}^{s} u_r y_{rp,3} \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ip,1} = 1, \\
& \sum_{r=1}^{s} u_r y_{rj,3} - \sum_{i=1}^{m} v_i x_{ij,1} \le 1, \quad j = 1,2,\ldots,n, \\
& v_i \ge 0, \qquad\qquad i = 1,2,\ldots,m, \\
& u_r \ge 0, \qquad\qquad r = 1,2,\ldots,s.
\end{aligned} \tag{5.49}$$

In sum the fuzzy efficiency of each DMU can be obtained by solving crisp linear models (5.47)–(5.49).

In matrix forms, the models (5.47)–(5.49) can be stated as follows:

$$\begin{aligned}
\max \quad & \left[0,\ldots,0,y_{1p,1},\ldots,y_{sp,1}\right]\left[v_1,\ldots,v_m,u_1,\ldots,u_s\right]^{\mathrm{t}} \\
\text{s.t.} \quad & \left[\begin{array}{ccc:ccc}
-x_{11,1} & \cdots & -x_{m1,1} & y_{11,3} & \cdots & y_{s1,3} \\
\vdots & & \vdots & \vdots & & \vdots \\
-x_{1n,1} & \cdots & -x_{mn,1} & y_{1n,3} & \cdots & y_{sn,3} \\
\hdashline
x_{1p,3} & \cdots & x_{mp,3} & 0 & \cdots & 0
\end{array}\right]
\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \end{array}\right]
\begin{array}{l} \le_n \\ =_1 \end{array}
\left[\begin{array}{c} 0_{n\times 1} \\ \hdashline 1 \end{array}\right]
\end{aligned} \tag{5.50}$$

$$\max \quad \left[0,\ldots,0,y_{1p,2},\ldots,y_{sp,2}\right]\left[v_1,\ldots,v_m,u_1,\ldots,u_s\right]^{\mathrm{t}}$$

$$\text{s.t.} \quad \left[\begin{array}{ccc|ccc} -x_{11,1} & \cdots & -x_{m1,1} & y_{11,3} & \cdots & y_{s1,3} \\ \vdots & & \vdots & \vdots & & \vdots \\ -x_{1n,1} & \cdots & -x_{mn,1} & y_{1n},3 & \cdots & y_{sn,3} \\ \hline x_{1p,2} & \cdots & x_{mp,2} & 0 & \cdots & 0 \end{array}\right]\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hline u_1 \\ \vdots \\ u_s \end{array}\right] \begin{array}{l} \leq_n \\ =_1 \end{array} \left[\begin{array}{c} 0_{n\times 1} \\ \hline 1 \end{array}\right] \tag{5.51}$$

$$\max \quad \left[0,\ldots,0,y_{1p,3},\ldots,y_{sp,3}\right]\left[v_1,\ldots,v_m,u_1,\ldots,u_s\right]^{\mathrm{t}}$$

$$\text{s.t.} \quad \left[\begin{array}{ccc|ccc} -x_{11,1} & \cdots & -x_{m1,1} & y_{11,3} & \cdots & y_{s1,3} \\ \vdots & & \vdots & \vdots & & \vdots \\ -x_{1n,1} & \cdots & -x_{mn,1} & y_{1n,3} & \cdots & y_{sn,3} \\ \hline x_{1p,1} & \cdots & x_{mp,1} & 0 & \cdots & 0 \end{array}\right]\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hline u_1 \\ \vdots \\ u_s \end{array}\right] \begin{array}{l} \leq_n \\ =_1 \end{array} \left[\begin{array}{c} 0_{n\times 1} \\ \hline 1 \end{array}\right] \tag{5.52}$$

The R code for model (5.50) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
df1 = data.frame(read_excel(path = "DN.L.xlsx", sheet="1"), check.names
= FALSE, fix.empty.names = TRUE);df1
df2 = data.frame(read_excel(path = "DN.M.xlsx", sheet="1"), check.names
= FALSE, fix.empty.names = TRUE);df2
df3 = data.frame(read_excel(path = "DN.U.xlsx", sheet="1"), check.names
= FALSE, fix.empty.names = TRUE)
m=2
s = ncol(df1)-m;s
N = nrow(df1);N
f.rhs = c(1, rep(0,N))
f.dir = c("=",rep("<=",N))
for (j in 1:N) {
f.obj = c((as.numeric(df1[j,(m+1):(m+s)])), rep(0,m))
con1= c(rep(0,s), (as.numeric(df3[j,1:m])))
con2= cbind(df3[,(m+1):(m+s)], -df1[,1:m])
```

```
f.con = rbind(con1, con2)
result = lp('max', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weight = result$solution
effvr = result$objval
} else {
weight = rbind(weight,result$solution)
effvr = rbind(effvr, result$objval) } }
EEs = data.frame(weight, effvr); EEs
colnames(EEs) = c('effi', rep("lambda", N),rep("x benchmark", m), rep("y
benchmark", s))
WriteXLS(EEs,"R.S.xls", row.names = TRUE, col.names = TRUE)
```

The R code for model (5.51) is as follows:

```
#required libraries
library(readxl)
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
df1 = data.frame(read_excel(path = "DN.L.xlsx", sheet="1"), check.names
= FALSE, fix.empty.names = TRUE)
df2 = data.frame(read_excel(path = "DN.M.xlsx", sheet="1"), check.names
= FALSE, fix.empty.names = TRUE)
df3 = data.frame(read_excel(path = "DN.U.xlsx", sheet="1"), check.names
= FALSE, fix.empty.names = TRUE)
m=2
s = ncol(df1)-m
N = nrow(df1)
f.rhs = c(1, rep(0,N))
f.dir = c("=",rep("<=",N))
for (j in 1:N) {
f.obj = c((as.numeric(df2[j,(m+1):(m+s)])), rep(0,m))
con1= c(rep(0,s), (as.numeric(df2[j,1:m])))
con2= cbind(df3[,(m+1):(m+s)], -df1[,1:m])
f.con = rbind(con1, con2)
result = lp('max', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weight = result$solution
effvr = result$objval
```

```
} else {
weight = rbind(weight,result$solution)
effvr = rbind(effvr, result$objval) } }
FEEs = data.frame(weight, effvr); FEEs
colnames(FEEs) = c('solution', "effi")
WriteXLS(FEEs,"result.xls", row.names = TRUE, col.names = TRUE)
```

The R code for model (5.52) is as follows:

```
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
df1 = data.frame(read_excel(path = "DN.L.xlsx", sheet="1"), check.names
= FALSE, fix.empty.names = TRUE);df1
df2 = data.frame(read_excel(path = "DN.M.xlsx", sheet="1"), check.names
= FALSE, fix.empty.names = TRUE);df2
df3 = data.frame(read_excel(path = "DN.U.xlsx", sheet="1"), check.names
= FALSE, fix.empty.names = TRUE)
m=2
s = ncol(df1)-m
N = nrow(df1)
f.rhs = c(1, rep(0,N))
f.dir = c("=",rep("<=",N))
for (j in 1:N) {
f.obj = c((as.numeric(df3[j,(m+1):(m+s)])), rep(0,m))
con1= c(rep(0,s), (as.numeric(df1[j,1:m])))
con2= cbind(df3[,(m+1):(m+s)], -df1[,1:m])
f.con = rbind(con1, con2)
result = lp('max', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weight = result$solution
effvr = result$objval
} else {
weight = rbind(weight,result$solution)
effvr = rbind(effvr, result$objval)
}
}
EEs = data.frame(weight, effvr); EEs
```

```
colnames(Es) = c('effi', rep("lambda",N),rep("x benchmark", m), rep("y benchmark", s))
WriteXLS(Es,"resultss.xls", row.names = TRUE, col.names = TRUE)
```

Example 5.8 Consider a fuzzy DEA efficiency evaluation problem with eight DMUs, each DMU with two fuzzy inputs and two fuzzy outputs. The data set is taken from Wang et al. [32] and is shown in Table 5.9.

The fuzzy efficiencies of these DMUs obtained by solving models (5.47)–(5.49) are given in Table 5.10.

From model (5.46), $\tilde{\theta}_j = \left(\theta_{j,1}, \theta_{j,2}, \theta_{j,3}\right) \preceq \tilde{1}$ is reduced to $\theta_{j,2} \leq 1$ and the following three crisp linear DEA models are solved to find the components of the fuzzy efficiency for each DMU:

$$\begin{aligned}
\max \quad & \theta_{p,1} = \sum_{r=1}^{s} u_r y_{rp,1} \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ip,3} = 1 \\
& \sum_{r=1}^{s} u_r y_{rj,2} - \sum_{i=1}^{m} v_i x_{ij,2} \leq 1, \quad j = 1, 2, \ldots, n, \\
& v_i \geq 0, \qquad\qquad\qquad\qquad i = 1, 2, \ldots, m, \\
& u_r \geq 0, \qquad\qquad\qquad\qquad r = 1, 2, \ldots, s.
\end{aligned} \tag{5.53}$$

$$\begin{aligned}
\max \quad & \theta_{p,2} = \sum_{r=1}^{s} u_r y_{rp,2} \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ip,2} = 1, \\
& \sum_{r=1}^{s} u_r y_{rj,2} - \sum_{i=1}^{m} v_i x_{ij,2} \leq 1, \quad j = 1, 2, \ldots, n, \\
& v_i \geq 0, \qquad\qquad\qquad\qquad i = 1, 2, \ldots, m, \\
& u_r \geq 0, \qquad\qquad\qquad\qquad r = 1, 2, \ldots, s.
\end{aligned} \tag{5.54}$$

Table 5.9 Eight DMUs with two fuzzy inputs and two fuzzy outputs

DMU	Input 1	Input 1	Output 1	Output 2
A	(2120, 2170, 2210)	(1870, 1870, 18,700	(14,500, 14,790, 14,860)	(3.1, 4.1, 4.9)
B	(1420, 1460, 1500)	(1340, 1340, 1340)	(12,470, 12,720, 12,790)	(1.2, 2.1, 3.0)
C	(1420, 1460, 1500)	(2360, 2360, 2360)	(17,900, 18,260, 18,400)	(3.3, 4.3, 5.0)
D	(1420, 1460, 1500)	(2020, 2020, 2020)	(14,970, 15,270, 15,400)	(2.7, 3.7, 4.6)
E	(1420, 1460, 1500)	(1550, 1550, 1550)	(13,980, 14,260, 14,330)	(1.0, 1.8, 2.7)
F	(1420, 1460, 1500)	(1760, 1760, 1760)	(14,030, 14,310, 14,400)	(1.6, 2.6, 3.6)
G	(2200, 2260, 2300)	(1980, 1980, 1980)	(16,540, 16,870, 17,000)	(2.4, 3.4, 4.4)
H	(2400, 2460, 2520)	(2250, 2250, 2250)	(17,600, 17,960, 18,100)	(2.6, 3.6, 4.6)

Table 5.10 Fuzzy efficiency of eight DMUs based on models (5.47)–5.49)

DMU	Efficiency
A	(0.8124, 0.9033, 1.0000)
B	(0.9750, 0.9945, 1.0000)
C	(0.7946, 0.8122, 0.9045)
D	(0.7764, 0.8050, 0.9070)
E	(0.9603, 0.9872, 1.0000)
F	(0.8352, 0.8518, 0.8852)
G	(0.8752, 0.8927, 0.9457)
H	(0.8195, 0.8363, 0.8864)

$$\begin{aligned}
\max \quad & \theta_{p,3} = \sum_{r=1}^{s} u_r y_{rp,3} \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ip,1} = 1, \\
& \sum_{r=1}^{s} u_r y_{rj,2} - \sum_{i=1}^{m} v_i x_{ij,2} \le 1, \quad j = 1,2,\ldots,n, \\
& v_i \ge 0, \qquad\qquad i = 1,2,\ldots,m, \\
& u_r \ge 0, \qquad\qquad r = 1,2,\ldots,s.
\end{aligned} \tag{5.55}$$

In sum the fuzzy efficiency of each DMU can be obtained by solving crisp linear models (5.53)–(5.55).

In matrix forms, models (5.53)–(5.55) can be stated as follows:

$$\begin{aligned}
\max \quad & \left[0,\ldots,0,y_{1p,1},\ldots,y_{sp,1}\right]\left[v_1,\ldots,v_m,u_1,\ldots,u_s\right]^{\mathrm{t}} \\
\text{s.t.} \quad & \left[\begin{array}{ccc:ccc}
-x_{11,2} & \cdots & -x_{m1,2} & y_{11,2} & \cdots & y_{s1,2} \\
\vdots & & \vdots & \vdots & & \vdots \\
-x_{1n,2} & \cdots & -x_{mn,2} & y_{1n,2} & \cdots & y_{sn,2} \\
\hdashline
x_{1p,3} & \cdots & x_{mp,3} & 0 & \cdots & 0
\end{array}\right]
\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \end{array}\right]
\begin{array}{l} \le_n \\ =_1 \end{array}
\left[\begin{array}{c} 0_{n\times 1} \\ \hdashline 1 \end{array}\right]
\end{aligned} \tag{5.56}$$

$$\begin{aligned}
\max \quad & \left[0,\ldots,0,y_{1p,2},\ldots,y_{sp,2}\right]\left[v_1,\ldots,v_m,u_1,\ldots,u_s\right]^{\mathrm{t}} \\
\text{s.t.} \quad & \left[\begin{array}{ccc:ccc}
-x_{11,2} & \cdots & -x_{m1,2} & y_{11,2} & \cdots & y_{s1,2} \\
\vdots & & \vdots & \vdots & & \vdots \\
-x_{1n,2} & \cdots & -x_{mn,2} & y_{1n,2} & \cdots & y_{sn,2} \\
\hdashline
x_{1p,2} & \cdots & x_{mp,2} & 0 & \cdots & 0
\end{array}\right]
\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \end{array}\right]
\begin{array}{l} \le_n \\ =_1 \end{array}
\left[\begin{array}{c} 0_{n\times 1} \\ \hdashline 1 \end{array}\right]
\end{aligned} \tag{5.57}$$

$$\begin{aligned}
&\max \quad \left[0,\ldots,0,y_{1p,3},\ldots,y_{sp,3}\right]\left[v_1,\ldots,v_m,u_1,\ldots,u_s\right]^{\mathrm{t}} \\
&\text{s.t.} \quad \left[\begin{array}{ccc:ccc}
-x_{11,2} & \cdots & -x_{m1,2} & y_{11,2} & \cdots & y_{s1,2} \\
\vdots & & \vdots & \vdots & & \vdots \\
-x_{1n,2} & \cdots & -x_{mn,2} & y_{1n,2} & \cdots & y_{sn,2} \\
\hdashline
x_{1p,1} & \cdots & x_{mp,1} & 0 & \cdots & 0
\end{array}\right]
\left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \end{array}\right]
\begin{array}{l} \leq_n \\ =_1 \end{array}
\left[\begin{array}{c} 0_{n\times 1} \\ \hdashline 1 \end{array}\right]
\end{aligned} \tag{5.58}$$

The R code for model (5.56) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data7.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE);df1
df2 = data.frame(read_excel(path = "data8.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE);df2
df3 = data.frame(read_excel(path = "data9.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df4 = data.frame(read_excel(path = "data10.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
#number of elements
m=4
s = ncol(df1)-m
N = nrow(df1)
#defining right hand side and directions
f.rhs = c(1, rep(0,N))
f.dir = c("=",rep("<=",N))
for (j in 1:N) {
#defining objective and matrix of constraints' coefficient
f.obj = c((as.numeric(df1[j,(m+1):(m+s)])), rep(0,m))
con1= c(rep(0,s), (as.numeric(df3[j,1:m])))
con2= cbind(df2[,(m+1):(m+s)], -df2[,1:m])
f.con = rbind(con1, con2)
#solving model
```

```
result = lp('max', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weight = result$solution
effvr = result$objval
} else {
weight = rbind(weight,result$solution)
effvr = rbind(effvr, result$objval) } }
EEs = data.frame(weight, effvr); EEs
colnames(Es) = c('effi', rep("lambda",N),rep("x benchmark", m), rep("y
benchmark", s))
WriteXLS(Es,"efficieny.xls", row.names = TRUE, col.names = TRUE)
```

The R code for model (5.57) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data7.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE);df1
df2 = data.frame(read_excel(path = "data8.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE);df2
df3 = data.frame(read_excel(path = "data9.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df4 = data.frame(read_excel(path = "data10.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
#number of elements
m=4
s = ncol(df1)-m
N = nrow(df1)
#defining right hand side and directions
f.rhs = c(1, rep(0,N))
f.dir = c("=",rep("<=",N))
for (j in 1:N) {
#defining objective and matrix of constraints' coefficient
f.obj = c((as.numeric(df2[j,(m+1):(m+s)])), rep(0,m))
con1= c(rep(0,s), (as.numeric(df2[j,1:m])))
```

```
con2= cbind(df2[,(m+1):(m+s)], -df2[,1:m])
f.con = rbind(con1, con2)
#solving model
result = lp('max', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weight = result$solution
effvr = result$objval
} else {
weight = rbind(weight,result$solution)
effvr = rbind(effvr, result$objval)} }
EEs = data.frame(weight, effvr); EEs
colnames(Es) = c('solution', "effi")
WriteXLS(Es,"4.32.xls", row.names = TRUE, col.names = TRUE)
```

The R code for model (5.58) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data7.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE);df1
df2 = data.frame(read_excel(path = "data8.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE);df2
df3 = data.frame(read_excel(path = "data9.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df4 = data.frame(read_excel(path = "data10.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
#number of elements
m=4
s = ncol(df1)-m
N = nrow(df1)
#defining right hand side and directions
f.rhs = c(1, rep(0,N))
f.dir = c("=",rep("<=",N))
for (j in 1:N) {
#defining objective and matrix of constraints' coefficient
```

```
f.obj = c((as.numeric(df3[j,(m+1):(m+s)])), rep(0,m))
con1= c(rep(0,s), (as.numeric(df1[j,1:m])))
con2= cbind(df2[,(m+1):(m+s)], -df2[,1:m])
f.con = rbind(con1, con2)
#solving model
result = lp('max', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weight = result$solution
effvr = result$objval
} else {
weight = rbind(weight,result$solution)
effvr = rbind(effvr, result$objval)} }
EEs = data.frame(weight, effvr); EEs
colnames(Es) = c('solution', "effi")
WriteXLS(Es,"4.33.xls", row.names = TRUE, col.names = TRUE)
```

Example 5.9 Consider the data given in Table 5.9. The fuzzy efficiencies of eight DMUs obtained by solving models (5.53)–(5.55) are gives in Table 5.11.

5.5.2 Bhardwaj et al.'s Approach

Bhardwaj et al. first [33] pointed out the flaws in the proposed methods of Wang et al. [32] and then proposed new method to resolve these flaws.

Since all models (5.47)–(5.49) and (5.52)–(5.55) are solved independently, so first the optimal weights obtained on solving these models, will not necessarily be same and second the restriction $\theta_{p,1} \leq \theta_{p,2} \leq \theta_{p,3}$ of the fuzzy efficiency $\tilde{\theta}_j = \left(\theta_{j,1}, \theta_{j,2}, \theta_{j,3}\right)$ may or may not be satisfied. To resolve these flaws they considered the following FDEA model:

Table 5.11 Fuzzy efficiency of eight DMUs based on models (5.53)–(5.55)

DMU	Efficiency
A	(0.8879, 1.0000, 1.2233)
B	(0.9830, 1.0000, 1.2121)
C	(0.8265, 0.9580, 1.0641)
D	(0.8020, 0.9061, 1.0585)
E	(0.9730, 1.0000, 1.1757)
F	(0.8398, 0.8819, 1.0056)
G	(0.8800, 0.9560, 1.0933)
H	(0.8240, 0.9115, 1.0570)

$$
\begin{aligned}
\max \quad & \tilde{\theta}_p = \left(\theta_{p,1}, \theta_{p,2}, \theta_{p,3}\right) \approx \left(\frac{\sum_{r=1}^{s} u_r y_{rp,1}}{\sum_{i=1}^{m} v_i x_{ip,3}}, \frac{\sum_{r=1}^{s} u_r y_{rp,2}}{\sum_{i=1}^{m} v_i x_{ip,2}}, \frac{\sum_{r=1}^{s} u_r y_{rp,3}}{\sum_{i=1}^{m} v_i x_{ip,1}}\right) \\
\text{s.t.} \quad & \left(\sum_{r=1}^{s} u_r y_{rj,1}, \sum_{r=1}^{s} u_r y_{rj,2}, \sum_{r=1}^{s} u_r y_{rj,3}\right) \\
& \preceq \left(\sum_{i=1}^{m} v_i x_{ij,1}, \sum_{i=1}^{m} v_i x_{ij,2}, \sum_{i=1}^{m} v_i x_{ij,3}\right), \quad \forall j \\
& v_i \geq 0, \quad i = 1, 2, \ldots, m, \\
& u_r \geq 0, \quad r = 1, 2, \ldots, s.
\end{aligned} \tag{5.59}
$$

They used the ranking method proposed by Ramik and Rimanek [25] given in Definition 5.5 to interpret the fuzzy inequalities of model (5.59) leading to the following model:

$$
\begin{aligned}
\max \quad & \tilde{\theta}_p = \left(\theta_{p,1}, \theta_{p,2}, \theta_{p,3}\right) \approx \left(\frac{\sum_{r=1}^{s} u_r y_{rp,1}}{\sum_{i=1}^{m} v_i x_{ip,3}}, \frac{\sum_{r=1}^{s} u_r y_{rp,2}}{\sum_{i=1}^{m} v_i x_{ip,2}}, \frac{\sum_{r=1}^{s} u_r y_{rp,3}}{\sum_{i=1}^{m} v_i x_{ip,1}}\right) \\
\text{s.t.} \quad & \textstyle\sum_{r=1}^{s} u_r y_{rj,1} \leq \sum_{i=1}^{m} v_i x_{ij,1}, \quad j = 1, 2, \ldots, n, \\
& \sum_{r=1}^{s} u_r y_{rj,2} \leq \sum_{i=1}^{m} v_i x_{ij,2}, \quad j = 1, 2, \ldots, n, \\
& \sum_{r=1}^{s} u_r y_{rj,3} \leq \sum_{i=1}^{m} v_i x_{ij,3}, \quad j = 1, 2, \ldots, n, \\
& v_i \geq 0, \qquad i = 1, 2, \ldots, m, \\
& u_r \geq 0, \qquad r = 1, 2, \ldots, s.
\end{aligned} \tag{5.60}
$$

Then they constructed the following three models to solve model (5.60):

$$
\begin{aligned}
\theta^*_{p,1} = \max \quad & \theta_{p,1} = \sum_{r=1}^{s} u_r y_{rp,1} \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ip,3} = 1, \\
& \sum_{r=1}^{s} u_r y_{rp,1} \leq \sum_{i=1}^{m} v_i x_{ip,3}, \qquad \forall j \\
& \text{Constraints of model (5.60).}
\end{aligned} \tag{5.61}
$$

$$
\begin{aligned}
\theta^*_{p,2} = \max \quad & \theta_{p,2} = \sum_{r=1}^{s} u_r y_{rp,2} \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ip,2} = 1, \\
& \sum_{r=1}^{s} u_r y_{rp,1} = \theta^*_{p,1} \sum_{i=1}^{m} v_i x_{ip,3}. \\
& \theta^*_{p,1} \sum_{i=1}^{m} v_i x_{ip,2} \leq \sum_{r=1}^{s} u_r y_{rp,2}, \qquad \forall j \\
& \text{Constraints of model (5.60).}
\end{aligned} \tag{5.62}
$$

$$
\begin{aligned}
\theta^*_{p,3} = \max \quad & \theta_{p,3} = \sum_{r=1}^{s} u_r y_{rp,3} \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i x_{ip,1} = 1, \\
& \sum_{r=1}^{s} u_r y_{rp,1} = \theta^*_{p,1} \sum_{i=1}^{m} v_i x_{ip,3}, \\
& \sum_{r=1}^{s} u_r y_{rp,2} = \theta^*_{p,2} \sum_{i=1}^{m} v_i x_{ip,2}, \\
& \theta^*_{p,2} \sum_{i=1}^{m} v_i x_{ip,1} \le \sum_{r=1}^{s} u_r y_{rp,3}, \qquad \forall j \\
& \sum_{r=1}^{s} u_r y_{rp,3} \le \sum_{i=1}^{m} v_i x_{ip,1}, \qquad \forall j \\
& \text{Constraints of model (5.60).}
\end{aligned}
\tag{5.63}
$$

By solving models (5.61)–(5.63), the fuzzy efficiency of DMU_p is given as $\tilde{\theta}^*_p = (\theta^*_{p,1}, \theta^*_{p,2}, \theta^*_{p,3})$.

In matrix forms, the models (5.61)–(5.63) can be stated as follows:

$$
\begin{aligned}
\max \quad & \begin{bmatrix} 0,\ldots,0, y_{1p,1},\ldots,y_{sp,1} \end{bmatrix} \begin{bmatrix} v_1,\ldots,v_m,u_1,\ldots,u_s \end{bmatrix}^{\mathrm{t}} \\
\text{s.t.} \quad & \left[\begin{array}{ccc:ccc}
-x_{11,3} & \cdots & -x_{m1,3} & y_{11,1} & \cdots & y_{s1,1} \\
\vdots & & \vdots & \vdots & & \vdots \\
-x_{1n,3} & \cdots & -x_{mn,3} & y_{1n,1} & \cdots & y_{sn,1} \\
\hdashline
x_{1p,3} & \cdots & x_{mp,3} & 0 & \cdots & 0
\end{array}\right]
\begin{bmatrix} v_1 \\ \vdots \\ v_m \\ u_1 \\ \vdots \\ u_s \end{bmatrix}
\begin{matrix} \le_n \\ =_1 \end{matrix}
\begin{bmatrix} 0_{n\times 1} \\ 1 \end{bmatrix}
\end{aligned}
\tag{5.64}
$$

$$
\begin{aligned}
\max \quad & \begin{bmatrix} 0,\ldots,0, y_{1p,2},\ldots,y_{sp,2} \end{bmatrix} \begin{bmatrix} v_1,\ldots,v_m,u_1,\ldots,u_s \end{bmatrix}^{\mathrm{t}} \\
\text{s.t.} \quad & \left[\begin{array}{ccc:ccc}
-\theta^*_{p,1}x_{11,3} & \cdots & -\theta^*_{p,1}x_{m1,3} & y_{11,1} & \cdots & y_{s1,1} \\
\vdots & & \vdots & \vdots & & \vdots \\
-\theta^*_{p,1}x_{1n,3} & \cdots & -\theta^*_{p,1}x_{mn,3} & y_{1n,1} & \cdots & y_{sn,1} \\
\hdashline
-\theta^*_{p,1}x_{11,2} & \cdots & -\theta^*_{p,1}x_{m1,2} & y_{11,2} & \cdots & y_{s1,2} \\
\vdots & & \vdots & \vdots & & \vdots \\
-\theta^*_{p,1}x_{1n,2} & \cdots & -\theta^*_{p,1}x_{mn,2} & y_{1n,2} & \cdots & y_{sn,2} \\
x_{1p,2} & \cdots & x_{mp,2} & 0 & \cdots & 0
\end{array}\right]
\begin{bmatrix} v_1 \\ \vdots \\ v_m \\ u_1 \\ \vdots \\ u_s \end{bmatrix}
\begin{matrix} \le_n \\ =_1 \end{matrix}
\begin{bmatrix} 0_{n\times 1} \\ 1 \end{bmatrix}
\end{aligned}
\tag{5.65}
$$

$$\max \quad \left[0,\ldots,0,y_{1p,2},\ldots,y_{sp,2}\right]\left[v_1,\ldots,v_m,u_1,\ldots,u_s\right]^{\mathrm{t}}$$

$$\text{s.t.} \quad \left[\begin{array}{ccc:ccc} -\theta^*_{p,1}x_{11,3} & \cdots & -\theta^*_{p,1}x_{m1,3} & y_{11,1} & \cdots & y_{s1,1} \\ \vdots & & \vdots & \vdots & & \vdots \\ -\theta^*_{p,1}x_{1n,3} & \cdots & -\theta^*_{p,1}x_{mn,3} & y_{1n,1} & \cdots & y_{sn,1} \\ \hdashline & & & & & \\ -\theta^*_{p,2}x_{11,2} & \cdots & -\theta^*_{p,2}x_{m1,2} & y_{11,2} & \cdots & y_{s1,2} \\ \vdots & & \vdots & \vdots & & \vdots \\ -\theta^*_{p,2}x_{1n,2} & \cdots & -\theta^*_{p,2}x_{mn,2} & y_{1n,2} & \cdots & y_{sn,2} \\ \hdashline -\theta^*_{p,2}x_{11,1} & \cdots & -\theta^*_{p,2}x_{m1,1} & y_{11,3} & \cdots & y_{s1,3} \\ \vdots & & \vdots & \vdots & & \vdots \\ -\theta^*_{p,2}x_{1n,1} & \cdots & -\theta^*_{p,2}x_{mn,1} & y_{1n,3} & \cdots & y_{sn,3} \\ x_{1p,1} & \cdots & x_{mp,1} & 0 & \cdots & 0 \end{array}\right] \left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \end{array}\right] \begin{array}{l} \leq_n \\ =_1 \end{array} \left[\begin{array}{c} 0_{n\times 1} \\ \hdashline 1 \end{array}\right] \tag{5.66}$$

Moreover, the matrix form of the constraints of model (5.60) is as follows:

$$\left[\begin{array}{ccc:ccc} -x_{11,1} & \cdots & -x_{m1,1} & y_{11,1} & \cdots & y_{s1,1} \\ \vdots & & \vdots & \vdots & & \vdots \\ -x_{1n,1} & \cdots & -x_{mn,1} & y_{1n,1} & \cdots & y_{sn,1} \\ \hdashline & & & & & \\ -x_{11,2} & \cdots & -x_{m1,2} & y_{11,2} & \cdots & y_{s1,2} \\ \vdots & & \vdots & \vdots & & \vdots \\ -x_{1n,2} & \cdots & -x_{mn,2} & y_{1n,2} & \cdots & y_{sn,2} \\ \hdashline -x_{11,3} & \cdots & -x_{m1,3} & y_{11,3} & \cdots & y_{s1,3} \\ \vdots & & \vdots & \vdots & & \vdots \\ -x_{1n,3} & \cdots & -x_{mn,3} & y_{1n,3} & \cdots & y_{sn,3} \end{array}\right] \tag{5.67}$$

Based on the given matrix forms in (5.64)–(5.67), the R code for model (5.64) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
```

```
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data7.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE);df1
df2 = data.frame(read_excel(path = "data8.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE);df2
df3 = data.frame(read_excel(path = "data9.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df4 = data.frame(read_excel(path = "data10.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
#number of elements
m=4
s = ncol(df1)-m
N = nrow(df1)
#defining right hand side and directions
f.rhs = c(1, 0, rep(0,3*N))
f.dir = c("=", "<=", rep("<=",3*N))
for (j in 1:N) {
#defining objective and matrix of constraints' coefficient
f.obj = c((as.numeric(df1[j,(m+1):(m+s)])), rep(0,m))
con1= c(rep(0,s), (as.numeric(df3[j,1:m])))
con2= c(as.numeric(df1[j,(m+1):(m+s)]), as.numeric(-df3[j,1:m]))
con3= cbind(df1[,(m+1):(m+s)], -df1[,1:m])
con4= cbind(df2[,(m+1):(m+s)], -df2[,1:m])
con5= cbind(df3[,(m+1):(m+s)], -df3[,1:m])
f.con = rbind(con1, con2, con3, con4, con5)
#solving model
result = lp('max', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weight = result$solution
effv = result$objval
} else {
weight = rbind(weight,result$solution)
effv = rbind(effv, result$objval)} }
EEs = data.frame(effv); EEs
colnames(Es) = c("solu', "effi")
WriteXLS(Es,"4.36.xls", row.names = TRUE, col.names = TRUE)
```

Also, the R code for model (5.65) is as follows. In this code the obtained optimal value of model (4.65) is read from an excel file named "tet1".

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data7.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE);df1
df2 = data.frame(read_excel(path = "data8.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE);df2
df3 = data.frame(read_excel(path = "data9.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df4 = data.frame(read_excel(path = "data10.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
tt = data.frame(read_excel(path = "tet1.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
#number of elements
m=4
s = ncol(df1)-m
N = nrow(df1)
#defining right hand side and directions
f.rhs = c(1, 0, 0, rep(0,3*N))
f.dir = c("=", "=", "<=", rep("<=",3*N))
for (j in 1:N) {
#defining objective and matrix of constraints coefficient
f.obj = c((as.numeric(df2[j,(m+1):(m+s)])), rep(0,m))
con1= c(rep(0,s), (as.numeric(df2[j,1:m])))
con2= c(as.numeric(df1[j,(m+1):(m+s)]), as.numeric(tt[j,1]) *as.numeric(-
df3[j,1:m]))
con3= c(as.numeric(-df2[j,(m+1):(m+s)]), as.numeric(tt[j,1]) *as.numeric
(df2[j,1:m]))
con4= cbind(df1[,(m+1):(m+s)], -df1[,1:m])
con5= cbind(df2[,(m+1):(m+s)], -df2[,1:m])
con6= cbind(df3[,(m+1):(m+s)], -df3[,1:m])
f.con = rbind(con1, con2, con3, con4, con5, con6)
#solving model
result = lp('max', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weight = result$solution
effv = result$objval
} else {
weight = rbind(weight,result$solution)
effv = rbind(effv, result$objval)} }
```

```
EEs = data.frame(weight, effv); EEs
colnames(EEs) = c('solution', "effi")
WriteXLS(EEs,"4.37.xls", row.names = TRUE, col.names = TRUE)
```

Finally, the R code of model (5.66) is as follows where the obtained optimal value of model (4.66) is gathered in excel file named "tet2".

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data7.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE);df1
df2 = data.frame(read_excel(path = "data8.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE);df2
df3 = data.frame(read_excel(path = "data9.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
df4 = data.frame(read_excel(path = "data10.xlsx", sheet="1"),
check.names = FALSE, fix.empty.names = TRUE)
tt = data.frame(read_excel(path = "tet1.xlsx", sheet="1"), check.names =
FALSE, fix.empty.names = TRUE)
#number of elements
m=4
s = ncol(df1)-m
N = nrow(df1)
#defining right hand side and direcctions
f.rhs = c(1, 0, 0, rep(0,3*N))
f.dir = c("=", "=", "<=", rep("<=",3*N))
for (j in 1:N) {
#defining objective and matrix of constraints' coefficient
f.obj = c((as.numeric(df2[j,(m+1):(m+s)])), rep(0,m))
con1= c(rep(0,s), (as.numeric(df2[j,1:m])))
con2= c(as.numeric(df1[j,(m+1):(m+s)]), as.numeric(tt[j,1]) *as.numeric(-
df3[j,1:m]))
con3= c(as.numeric(-df2[j,(m+1):(m+s)]), as.numeric(tt[j,1]) *as.numeric
(df2[j,1:m]))
con4= cbind(df1[,(m+1):(m+s)], -df1[,1:m])
con5= cbind(df2[,(m+1):(m+s)], -df2[,1:m])
con6= cbind(df3[,(m+1):(m+s)], -df3[,1:m])
```

```
f.con = rbind(con1, con2, con3, con4, con5, con6)
#solving model
result = lp('max', f.obj, f.con, f.dir, f.rhs, scale=0, compute.sens=F)
if (j==1) {
weight = result$solution
effv = result$objval
} else {
weight = rbind(weight,result$solution)
effv = rbind(effv, result$objval)} }
EEs = data.frame(weight, effv); EEs
colnames(EEs) = c('solution', "effi")
WriteXLS(EEs,"4.37.xls", row.names = TRUE, col.names = TRUE)
```

Example 5.10 Consider the data given in Table 5.9. The resulting fuzzy efficiency scores are given in Table 5.12. The DMUs can be ranked according to their degree of preferences by using the Wang et al.'s ranking approach [32].

5.5.3 Azar et al.'s Approach

The idea behind the proposed method of Azar et al.'s [34] is to use the common set of weights (CSW) to find the fuzzy efficiency of DMUs based upon fuzzy arithmetic operations.

They proposed following additive model to maximize the efficiency of all DMUs (by combining both the input- and output-oriented), with the condition that efficiency ratio for all units are being less than or equal to one:

Table 5.12 Fuzzy efficiency of eight DMUs

DMU	Efficiency
A	(0.812, 0.829, 0.833)
B	(0.975, 1.000, 1.000)
C	(0.797, 0.815, 0.825)
D	(0.776, 0.792, 0.799)
E	(0.973, 1.000, 1.000)
F	(0.835, 0.852, 0.857)
G	(0.875, 0.893, 0.900)
H	(0.820, 0.836, 0.843)

$$
\begin{aligned}
\max \quad & \min_{j=1,\ldots,n}\left\{\sum_{r=1}^{s} u_r y_{rj} - \sum_{i=1}^{m} v_i x_{ij}\right\} \\
\text{s.t.} \quad & \frac{\sum_{r=1}^{s} u_r y_{rj}}{\sum_{i=1}^{m} v_i x_{ij}} \le 1, \\
& \sum_{r=1}^{s} u_r \ge 1, \qquad j = 1,2,\ldots,n, \\
& \sum_{i=1}^{m} v_i \ge 1, \\
& v_i \ge \varepsilon, \qquad i = 1,2,\ldots,m, \\
& u_r \ge \varepsilon, \qquad r = 1,2,\ldots,s.
\end{aligned}
\tag{5.68}
$$

When the input and output data are assumed to be triangular fuzzy numbers, the model (5.68) can be written as follows:

$$
\begin{aligned}
\max \quad & \min_{j=1,\ldots,n}\left\{\left(\sum_{r=1}^{s} u_r y_{rj,1-} - \sum_{i=1}^{m} v_i x_{ij,3}, \sum_{r=1}^{s} u_r y_{rj,2}\right.\right. \\
& \left.\left. - \sum_{i=1}^{m} v_i x_{ij,2}, \sum_{r=1}^{s} u_r y_{rj,3-} - \sum_{i=1}^{m} v_i x_{ij,1}\right)\right\} \\
\text{s.t.} \quad & \left(\frac{\sum_{r=1}^{s} u_r y_{rj,1}}{\sum_{i=1}^{m} v_i x_{ij,3}}, \frac{\sum_{r=1}^{s} u_r y_{rj,2}}{\sum_{i=1}^{m} v_i x_{ij,2}}, \frac{\sum_{r=1}^{s} u_r y_{rj,3}}{\sum_{i=1}^{m} v_i x_{ij,1}}\right) \le 1, \qquad \forall j, \\
& \sum_{r=1}^{s} u_r \ge 1, \\
& \sum_{i=1}^{m} v_i \ge 1, \\
& v_i \ge \varepsilon, \qquad i = 1,2,\ldots,m, \\
& u_r \ge \varepsilon, \qquad r = 1,2,\ldots,s.
\end{aligned}
\tag{5.69}
$$

Similar to first approach of Wang et al.'s [34], the model (5.69) is simplified to the following linear programming model:

$$
\begin{aligned}
\max \quad & \theta \\
\text{s.t.} \quad & \sum_{r=1}^{s} u_r y_{rj,1-} - \sum_{i=1}^{m} v_i x_{ij,3} \ge \theta, \qquad \forall j \\
& \sum_{r=1}^{s} u_r y_{rj,3} - \sum_{i=1}^{m} v_i x_{ij,1} \le 0, \qquad \forall j \\
& \sum_{r=1}^{s} u_r \ge 1, \\
& \sum_{i=1}^{m} v_i \ge 1, \\
& v_i \ge \varepsilon, \qquad i = 1,2,\ldots,m, \\
& u_r \ge \varepsilon, \qquad r = 1,2,\ldots,s.
\end{aligned}
\tag{5.70}
$$

By solving model (5.70) and finding the optimal weights the fuzzy efficiency of each DMU is given as follows:

$$\tilde{\theta}_j^* = \frac{\sum_{r=1}^{s} u_r^* \tilde{y}_{rj}}{\sum_{i=1}^{m} v_i^* \tilde{x}_{ij}} = \left(\frac{\sum_{r=1}^{s} u_r^* y_{rj,1}}{\sum_{i=1}^{m} v_i^* x_{ij,3}}, \frac{\sum_{r=1}^{s} u_r^* y_{rj,2}}{\sum_{i=1}^{m} v_i^* x_{ij,2}}, \frac{\sum_{r=1}^{s} u_r^* y_{rj,3}}{\sum_{i=1}^{m} v_i^* x_{ij,1}} \right) \tag{5.71}$$

In matrix form, the model (5.70) can be stated as follows:

$$\begin{aligned} \max \quad & [0,\ldots,0,0,\ldots,0,1] \left[v_1,\ldots,v_m,u_1,\ldots,u_s,\bar{f} \right]^{\mathrm{t}} \\ \text{s.t.} \quad & \left[\begin{array}{ccc:ccc:c} -x_{11,3} & \cdots & -x_{m1,3} & y_{11,1} & \cdots & y_{s1,1} & -1 \\ \vdots & & \vdots & \vdots & & \vdots & 0 \\ -x_{1n,3} & \cdots & -x_{mn,3} & y_{1n,1} & \cdots & y_{sn,1} & 0 \\ \hdashline -x_{11,1} & \cdots & -x_{m1,1} & y_{11,3} & \cdots & y_{s1,3} & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ -x_{1n,1} & \cdots & -x_{mn,1} & y_{1n,3} & \cdots & y_{sn,3} & 0 \\ \hdashline 1 & \cdots & 1 & 0 & \cdots & 0 & 0 \\ \hdashline 0 & \cdots & 0 & 1 & \cdots & 1 & 0 \\ 1 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 & 0 \\ \hdashline 0 & \cdots & 0 & 1 & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \\ 0 & \cdots & 0 & 0 & \cdots & 1 & \end{array}\right] \left[\begin{array}{c} v_1 \\ \vdots \\ v_m \\ \hdashline u_1 \\ \vdots \\ u_s \\ \bar{f} \end{array}\right] \left[\begin{array}{c} \geq \\ \leq \\ \geq \\ \hdashline \geq \\ \geq \\ \geq \\ \geq \end{array}\right] \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ \hdashline 1 \\ \varepsilon \\ \varepsilon \end{array}\right] \end{aligned} \tag{5.72}$$

The R code of model (5.72) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data7.xlsx", sheet = "1"),
check.names = FALSE, fix.empty.names = TRUE)
df2 = data.frame(read_excel(path = "data8.xlsx", sheet = "1"),
check.names = FALSE, fix.empty.names = TRUE)
df3 = data.frame(read_excel(path = "data9.xlsx", sheet = "1"),
check.names = FALSE, fix.empty.names = TRUE)
```

```
df4 = data.frame(read_excel(path = "data10.xlsx", sheet = "1"),
check.names = FALSE, fix.empty.names = TRUE)
#parameter and number of elements
Eps = 0.1
m = 4
s = ncol(df1)-m
N = nrow(df1)
#defining right hand side, direction, and objective
f.rhs = c(rep(0,N), rep(0,N), 1, 1, Eps, Eps)
f.dir = c(rep(" >=",N), rep(" <=",N), " >=", " >=", " >=", " >=")
f.obj = c(rep(0,s), rep(0,m), 1, -1)
#defining matrix of constraints' coefficient
con1 = cbind(df1[,(m + 1):(m + s)], -df3[,1:m], matrix(-1,N,1), matrix(1,N,1))
con2 = cbind(df3[,(m + 1):(m + s)], -df1[,1:m], matrix(0,N,1), matrix(0,N,1))
con3 = cbind(diag(s), matrix(0,s,m), matrix(0,s,1), matrix(0,s,1))
con4 = cbind(matrix(0,m,s), diag(m), matrix(0,m,1), matrix(0,m,1))
con5 = c(rep(1,s), rep(0,m), 0, 0)
con6 = c(rep(0,s), rep(1,m), 0, 0)
f1 = rbind(as.matrix(con1), as.matrix(con2))
f2 = rbind(con5, con6)
f3 = rbind(as.matrix(con3), as.matrix(con4))
f.con = rbind(unname(f1), unname(f2), unname(f3))
#solving model
result = lp('max', f.obj, f.con, f.dir, f.rhs, scale = 0, compute.sens = F)
weight = result$solution; weight
objective = result$objval; objective
WriteXLS(weight,"4.41.xls", row.names = TRUE, col.names = TRUE)
```

Example 5.10 Consider the data given in Table 5.9. The resulting fuzz efficiencies of DMUs under consideration are given in Table 5.13. The DMUs can be ranked according to their degree of preferences by using the Azar et al.'s ranking approach [34] using the average of possibility of dominance.

Table 5.13 Fuzzy efficiency of Eight DMUs

DMU	Efficiency
A	(0.8026, 0.8208, 0.8272)
B	(0.9682, 0.9911, 1.0000)
C	(0.7908, 0.8084, 0.8169)
D	(0.7664, 0.7841, 0.7931)
E	(0.9488, 0.9707, 0.9786)
F	(0.8235, 0.8437, 0.8514)
G	(0.8659, 0.8853, 0.8953)
H	(0.8142, 0.8334, 0.8425)

It should be noted that Azar et al.'s [34] have used the rakig approaches given in Definitions 1.25 and 1.26 ro provide a full ranking of all DMUs based on obtained fuzzy efficiency scores given in Table 5.13.

5.5.4 The MOLP Approach

The main idea behind of this approach is to convert the fully fuzzy DEA model into a MOLP problem and to use the lexicographic technique for solving the resulting model.

Hatami-marbini et al. [35] formulated fully fuzzy version of input-oriented CCR envelopment model as follows:

$$\begin{aligned}
\min \quad & \tilde{\theta}_p \\
\text{s.t.} \quad & \sum_{j=1}^{n} \tilde{\lambda}_j \otimes \tilde{x}_{ij} + \tilde{s}_i^- = \tilde{\theta}_p^{CRS} \otimes \tilde{x}_{ip}, && i = 1, 2, \ldots, m, \\
& \sum_{j=1}^{n} \tilde{\lambda}_j \otimes \tilde{y}_{rj} = \tilde{s}_r^+ + \tilde{y}_{rp}, && r = 1, 2, \ldots, s, \\
& \tilde{\lambda}_j \succsim \tilde{0}, && j = 1, 2, \ldots, n, \\
& \tilde{s}_i^- \succsim \tilde{0}, && i = 1, 2, \ldots, m, \\
& \tilde{s}_r^+ \succsim \tilde{0}, && r = 1, 2, \ldots, s.
\end{aligned} \tag{5.73}$$

When all parameters of model (5.72) are assumed to be trapezoidal fuzzy numbers, the model can be written as follows:

$$\begin{aligned}
\min \quad & \left(\theta_{p,1}^{CRS}, \theta_{p,2}^{CRS}, \theta_{p,3}^{CRS}, \theta_{p,4}^{CRS}\right) \\
& \text{s.t.} \\
& \sum_{j=1}^{n} \left(\lambda_{j,1}, \lambda_{j,2}, \lambda_{j,3}, \lambda_{j,4}\right) \otimes \left(x_{ij,1}, x_{ij,2}, x_{ij,3}, x_{ij,4}\right) + \left(s_{i,1}^-, s_{i,2}^-, s_{i,3}^-, s_{i,4}^-\right) \\
& \quad = \left(\theta_{p,1}^{CRS}, \theta_{p,2}^{CRS}, \theta_{p,3}^{CRS}, \theta_{p,4}^{CRS}\right) \otimes \left(x_{ip,1}, x_{ip,2}, x_{ip,3}, x_{ip,4}\right), && \forall i \\
& \sum_{j=1}^{n} \left(\lambda_{j,1}, \lambda_{j,2}, \lambda_{j,3}, \lambda_{j,4}\right) \otimes \left(y_{rj,1}, y_{rj,2}, y_{rj,3}, y_{rj,4}\right) \\
& \quad = \left(s_{r,1}^+, s_{r,2}^+, s_{r,3}^+, s_{r,4}^+\right) + \left(y_{rp,1}, y_{rp,2}, y_{rp,3}, y_{rp,4}\right), && \forall r, \\
& \left(\lambda_{j,1}, \lambda_{j,2}, \lambda_{j,3}, \lambda_{j,4}\right), \left(s_{i,1}^-, s_{i,2}^-, s_{i,3}^-, s_{i,4}^-\right), \left(s_{r,1}^+, s_{r,2}^+, s_{r,3}^+, s_{r,4}^+\right) \in TF(\mathbb{R})^+, && \forall i, j, r.
\end{aligned} \tag{5.74}$$

Based upon fuzzy arithmetic operations model (5.74) can be reformulated as follows:

$$
\begin{aligned}
\min \quad & \left(\theta_{p,1}^{CRS}, \theta_{p,2}^{CRS}, \theta_{p,3}^{CRS}, \theta_{p,4}^{CRS}\right) \\
\text{s.t.} \quad & \sum_{j=1}^{n} \lambda_{j,k} x_{ij,k} + s_{i,k}^{-} = \theta_{p,k}^{CRS} x_{ip,k}, \quad i = 1, 2, \ldots, m, \ k = 1, 2, 3, 4, \\
& \sum_{j=1}^{n} \lambda_{j,k} y_{rj,k} = s_{r,k}^{+} + y_{rp,k}, \quad r = 1, 2, \ldots, s, \ k = 1, 2, 3, 4, \\
& \lambda_{j,1} \geq 0; \ \lambda_{j,2} - \lambda_{j,1} \geq 0; \ \lambda_{j,3} - \lambda_{j,2} \geq 0; \ \lambda_{j,4} - \lambda_{j,3} \geq 0, \qquad \forall j \\
& s_{i,1}^{-} \geq 0; \ s_{i,2}^{-} - s_{i,1}^{-} \geq 0; \ s_{i,3}^{-} - s_{i,2}^{-} \geq 0; \ s_{i,4}^{-} - s_{i,3}^{-} \geq 0, \qquad \forall i \\
& s_{r,1}^{+} \geq 0; \ s_{r,2}^{+} - s_{r,1}^{+} \geq 0; \ s_{r,3}^{+} - s_{r,2}^{+} \geq 0; s_{r,4}^{+} - s_{r,3}^{+} \geq 0, \qquad \forall r \\
& \theta_{p,2}^{CRS} - \theta_{p,1}^{CRS} \geq 0, \ \theta_{p,3}^{CRS} - \theta_{p,2}^{CRS} \geq 0, \ \theta_{p,4}^{CRS} - \theta_{p,3}^{CRS} \geq 0.
\end{aligned} \tag{5.75}
$$

Hatami-marbini et al. [35] proposed the following the following linear programming models derived from the lexicographic multi-objective linear programming (MOLP) approach:

$$
\begin{aligned}
\theta_{p,4}^{CRS,*} = \min \quad & \theta_{p,4}^{CRS} \\
\text{s.t.} \quad & \theta_{p,2}^{CRS} - \theta_{p,1}^{CRS} \geq 0, \ \theta_{p,3}^{CRS} - \theta_{p,2}^{CRS} \geq 0, \ \theta_{p,4}^{CRS} - \theta_{p,3}^{CRS} \geq 0, \\
& \text{Remaining constraints of the model (5.75).}
\end{aligned} \tag{5.76}
$$

$$
\begin{aligned}
\theta_{p,3}^{CRS,*} = \min \quad & \theta_{p,3}^{CRS} \\
\text{s.t.} \quad & \theta_{p,4}^{CRS} = \theta_{p,4}^{CRS,*} \\
& \theta_{p,2}^{CRS} - \theta_{p,1}^{CRS} \geq 0, \ \theta_{p,3}^{CRS} - \theta_{p,2}^{CRS} \geq 0, \ \theta_{p,4}^{CRS,*} - \theta_{p,3}^{CRS} \geq 0, \\
& \text{Remaining constraints of the model (5.75).}
\end{aligned} \tag{5.77}
$$

$$
\begin{aligned}
\theta_{p,2}^{CRS,*} = \min \quad & \theta_{p,2}^{CRS} \\
\text{s.t.} \quad & \theta_{p,4}^{CRS} = \theta_{p,4}^{CRS,*} \\
& \theta_{p,3}^{CRS} = \theta_{p,3}^{CRS,*} \\
& \theta_{p,2}^{CRS} - \theta_{p,1}^{CRS} \geq 0, \ \theta_{p,3}^{CRS,*} - \theta_{p,2}^{CRS} \geq 0 \\
& \text{Remaining constraints of the model (5.75).}
\end{aligned} \tag{5.78}
$$

$$
\begin{aligned}
\theta_{p,1}^{CRS,*} = \min \quad & \theta_{p,1}^{CRS} \\
\text{s.t.} \quad & \theta_{p,4}^{CRS} = \theta_{p,4}^{CRS,*} \\
& \theta_{p,3}^{CRS} = \theta_{p,3}^{CRS,*} \\
& \theta_{p,2}^{CRS} = \theta_{p,1}^{CRS,*} \\
& \theta_{p,2}^{CRS,*} - \theta_{p,1}^{CRS} \geq 0, \\
& \text{Remaining constraints of the model (5.75).}
\end{aligned} \tag{5.79}
$$

By solving models (5.76)–(5.79) the relative fuzzy efficiency of DMU_p is given as $\tilde{\theta}_p^{CRS,*} = \left(\theta_{p,1}^{CRS,*}, \theta_{p,2}^{CRS,*}, \theta_{p,3}^{CRS,*}, \theta_{p,4}^{CRS,*}\right)$. Analogously to the crisp CCR model, the following Phase II model can be used to maximize the sum of fuzzy input excesses and fuzzy output shortfalls, while preserving $\tilde{\theta}_p^{CRS,*}$:

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{m}\left(s_{i,1}^{-}+s_{i,2}^{-}+s_{i,3}^{-}+s_{i,4}^{-}\right)+\sum_{r=1}^{s}\left(s_{r,1}^{+}+s_{r,2}^{+}+s_{r,3}^{+}+s_{r,4}^{+}\right)\\
\text{s.t.} \quad & \sum_{j=1}^{n}\lambda_{j,k}x_{ij,k}+s_{i,k}^{-}=\theta_{p,k}^{CRS,*}x_{ip,k}, \quad i=1,2,\ldots,m, \quad k=1,2,3,4,\\
& \sum_{j=1}^{n}\lambda_{j,k}y_{rj,k}=s_{r,k}^{+}+y_{rp,k}, \quad r=1,2,\ldots,s, \quad k=1,2,3,4,\\
& \lambda_{j,1}\geq 0;\ \lambda_{j,2}-\lambda_{j,1}\geq 0;\ \lambda_{j,3}-\lambda_{j,2}\geq 0;\ \lambda_{j,4}-\lambda_{j,3}\geq 0; \quad j=1,2,\ldots,n,\\
& s_{i,1}^{-}\geq 0;\ s_{i,2}^{-}-s_{i,1}^{-}\geq 0;\ s_{i,3}^{-}-s_{i,2}^{-}\geq 0;\ s_{i,4}^{-}-s_{i,3}^{-}\geq 0, \quad i=1,2,\ldots,m,\\
& s_{r,1}^{+}\geq 0;\ s_{r,2}^{+}-s_{r,1}^{+}\geq 0;\ s_{r,3}^{+}-s_{r,2}^{+}\geq 0;\ s_{r,4}^{+}-s_{r,3}^{+}\geq 0, \quad r=1,2,\ldots,s.
\end{aligned}
\tag{5.80}
$$

Depending of the optimal solutions of models (5.76)–(5.80), all DMUs are classified into three groups based on the following definitions:

Definition 5.10 Let $\tilde{\theta}_p^{CRS,*}=\left(\theta_{p,1}^{CRS,*},\theta_{p,2}^{CRS,*},\theta_{p,3}^{CRS,*},\theta_{p,4}^{CRS,*}\right)$ is the relative fuzzy efficiency of DMU_p and $\tilde{s}_i^{-*}=\left(s_{i,1}^{-*},s_{i,2}^{-*},s_{i,3}^{-*},s_{i,4}^{-*}\right)$ and $\tilde{s}_r^{+*}=\left(s_{r,1}^{+*},s_{r,2}^{+*},s_{r,3}^{+*},s_{r,4}^{+*}\right)$ the optimal solution of model (5.80), then DMU_p can be classified into one of these three groups:

- DMU_p is "*fully FDEA efficient*" if and only if $\tilde{\theta}_p^{CRS,*}=(1,1,1,1)$, $\tilde{s}_i^{-*}=(0,0,0,0)\ \forall i$ and $\tilde{s}_r^{+*}=(0,0,0,0)\ \forall r$.
- DMU_p is called "*fully FDEA weakly efficient*" if and only if $\tilde{\theta}_p^{CRS,*}=(1,1,1,1)$ and the optimal value of model (5.80) is positive (i.e. $\sum_{i=1}^{m}\tilde{s}_i^{-*}+\sum_{r=1}^{s}\tilde{s}_r^{+*}\neq(0,0,0,0)$).
- DMU_p is called "*fully FDEA inefficient*" if and only if $\tilde{\theta}_p^{CRS,*}\neq(1,1,1,1)$.

By using the matrix form of constraints of model (5.75), the matrix form of model (5.76) can be stated as follows:

$$
\left[\begin{array}{c|ccc|ccc|ccc}
-x_{1p,k} & x_{11,k} & \cdots & x_{1n,k} & 1 & \cdots & 0 & 0 & \cdots & 0\\
\vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots\\
-x_{mp,k} & x_{m1,k} & \cdots & x_{mn,k} & 0 & \cdots & 1 & 0 & \cdots & 0\\
\hline
0 & y_{11,k} & \cdots & y_{1n,k} & 0 & \cdots & 0 & -1 & \cdots & 0\\
\vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots\\
0 & y_{s1,k} & \cdots & y_{sn,k} & 0 & \cdots & 0 & 0 & \cdots & -1
\end{array}\right]
\left[\begin{array}{c}\theta\\ \hline \lambda\\ S^{-}\\ S^{+}\end{array}\right]
=\left[\begin{array}{c}0_{m\times 1}\\ \hline y_{1p,k}\\ \vdots\\ y_{sp,k}\end{array}\right]
\quad k=1,2,3,4
\tag{5.81}
$$

The R code for model (5.76) is as follows:

```
#required libraries
library(readxl)
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data71.xlsx", sheet="1"))
df2 = data.frame(read_excel(path = "data81.xlsx", sheet="1"))
df3 = data.frame(read_excel(path = "data81.xlsx", sheet="1"))
df4 = data.frame(read_excel(path = "data101.xlsx", sheet="1"))
#defining inputs and outputs
inputs1=data.frame(df1[1:4])
outputs1=data.frame(df1[5:8])
inputs2=data.frame(df2[1:4])
outputs2=data.frame(df2[5:8])
inputs3=data.frame(df3[1:4])
outputs3=data.frame(df3[5:8])
inputs4=data.frame(df4[1:4])
outputs4=data.frame(df4[5:8])
#number of elements
n=dim(df1)[1]
m=dim(inputs1)[2]
s=dim(outputs1)[2]
#defining directions, objective, and right hand side
f.dir=c(rep("=",4*(m+s)), rep(">=",4*(N+m+s)+3))
for (j in 1:n)
{
f.obj=c(0,rep(0,m+s+n), 0,rep(0,m+s+n), 0,rep(0,m+s+n), 1, rep(0,m+s+n))
f.rhs=c(rep(0,m),as.numeric(outputs1[j,]), rep(0,m),as.numeric(outputs2[j,]),
rep(0,m),as.numeric(outputs3[j,]),rep(0,m),as.numeric(outputs4[j,]),      rep
(0,4*(N+m+s)+3))
#defining matrix of constrraints' coefficient
con1.1=rbind(t(inputs1),t(outputs1))
con2.1=rbind(diag(m),matrix(0,s,m))
con3.1=rbind(matrix(0,m,s),-diag(s))
con4.1=rbind(t(-inputs1[j,]),matrix(0,s,1))
con1.2=rbind(t(inputs2),t(outputs2))
con2.2=rbind(diag(m),matrix(0,s,m))
con3.2=rbind(matrix(0,m,s),-diag(s))
con4.2=rbind(t(-inputs2[j,]),matrix(0,s,1))
con1.3=rbind(t(inputs3),t(outputs3))
con2.3=rbind(diag(m),matrix(0,s,m))
```

```
con3.3=rbind(matrix(0,m,s),-diag(s))
con4.3=rbind(t(-inputs3[j,]),matrix(0,s,1))
con1.4=rbind(t(inputs4),t(outputs4))
con2.4=rbind(diag(m),matrix(0,s,m))
con3.4=rbind(matrix(0,m,s),-diag(s))
con4.4=rbind(t(-inputs4[j,]),matrix(0,s,1))
f.con1=cbind(con4.1,con1.1,con2.1,con3.1)
f.con2=cbind(con4.2,con1.2,con2.2,con3.2)
f.con3=cbind(con4.3,con1.3,con2.3,con3.3)
f.con4=cbind(con4.4,con1.4,con2.4,con3.4)
f.con1.1=cbind(f.con1,matrix(0, (m+s), (3*(m+s+N+1))))
f.con2.1=cbind(matrix(0, (m+s), (m+s+N+1)), f.con2,matrix(0, (m+s), (2*(m
+s+N+1))))
f.con3.1=cbind(matrix(0, (m+s), 2*(m+s+N+1)), f.con3,matrix(0, (m+s),
(m+s+N+1)))
f.con4.1=cbind(matrix(0, (m+s), 3*(m+s+N+1)), f.con4)
f.con.1 = rbind(f.con1.1, f.con2.1, f.con3.1, f.con4.1)
c.con1= cbind(matrix(0,N,1), matrix(1,N,1), matrix(0,N,m+s+N-1),
matrix(0,N,3*(m+s+N+1)))
c.con2= cbind(matrix(0,N,1), matrix(-1,N,1), matrix(0,N,m+s+N), matrix(1,N,1),
matrix(0,N,m+s+N-1),matrix(0,N,2*(m+s+n+1)))
c.con3= cbind(matrix(0,N,N+m+s+1), matrix(0,N,1),
matrix(-1,N,1),matrix(0,N,m+s+N),matrix(1,N,1),matrix(0,N,m+s
+N-1),matrix(0,N,(m+s+n+1)))
c.con4= cbind(matrix(0,N,2*(N+m+s+1)), matrix(0,N,1),
matrix(-1,N,1),matrix(0,N,m+s+N),matrix(1,N,1),matrix(0,N,m+s+N-1))
f.con.2=rbind(c.con1,c.con2,c.con3,c.con4)
a.con1 = cbind(matrix(0,m,N+1), matrix(1,m,1),matrix(0,m,m+s-1),
matrix(0,m,3*(m+s+N+1)))
a.con2 = cbind(matrix(0,m,N+1), matrix(-1,m,1),matrix(0,m,m+s+N),
matrix(1,m,1), matrix(0,m,m+s-1),matrix(0,m,2*(m+s+N+1)))
a.con3 = cbind(matrix(0,m,(m+s+N+1)), matrix(0,m,N+1), matrix(-1,m,1),
matrix(0,m,m+s+N), matrix(1,m,1), matrix(0,m,m+s-1),matrix(0,m,(m+s+N+1)))
a.con4 = cbind(matrix(0,m,2*(m+s+N+1)), matrix(0,m,N+1), matrix(-1,
m,1), matrix(0,m,m+s+N), matrix(1,m,1), matrix(0,m,m+s-1))
f.con.3=rbind(a.con1,a.con2,a.con3,a.con4)
b.con1 = cbind(matrix(0,s,N+1+m), matrix(1,s,1),matrix(0,s,s-1),
matrix(0,m,3*(m+s+N+1)))
b.con2 = cbind(matrix(0,s,N+1+m), matrix(-1,s,1),matrix(0,s,m+s+N),
matrix(1,s,1), matrix(0,s,s-1),matrix(0,m,2*(m+s+N+1)))
b.con3 = cbind(matrix(0,s,(m+s+N+1)), matrix(0,s,N+1+m), matrix(-1,s,1),
```

```
matrix(0,m,m+s+N), matrix(1,s,1), matrix(0,s,s-1),matrix(0,s,(m+s+N+1)))
b.con4 = cbind(matrix(0,m,2*(m+s+N+1)), matrix(0,s,N+1+m), matrix(-1,
s,1), matrix(0,s,m+s+N), matrix(1,s,1), matrix(0,s,s-1))
f.con.4=rbind(b.con1,b.con2,b.con3,b.con4)
d.con1= c(-1, rep(0,N+m+s), 1,rep(0,N+m+s), rep(0, 2*(N+m+s+1)))
d.con2= c(rep(0, (N+m+s+1)), -1, rep(0,N+m+s), 1,rep(0,N+m+s), rep(0,
(N+m+s+1)))
d.con3= c(rep(0, 2*(N+m+s+1)), -1, rep(0,N+m+s), 1,rep(0,N+m+s))
f.con.5= rbind(d.con1, d.con2, d.con3)
f.con= rbind(f.con.1, f.con.2, f.con.3, f.con.4, f.con.5)
results=lp ("min",f.obj, f.con, f.dir, f.rhs)
if (j==1)
{
effcrs = results$objval
}
else
{
effcrs = rbind(effcrs, results$objval)
}
}
EEs = data.frame(effcrs)
colnames(EEs) = c('effi');EEs
WriteXLS(EEs,"4.46.xls", row.names = F, col.names = F)
```

Assuming the optimal value of model (5.76) is teta4, the R code for model (5.77) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data71.xlsx", sheet="1"))
df2 = data.frame(read_excel(path = "data81.xlsx", sheet="1"))
df3 = data.frame(read_excel(path = "data81.xlsx", sheet="1"))
df4 = data.frame(read_excel(path = "data101.xlsx", sheet="1"))
teta4 = data.frame(read_excel(path = "teta4-(46).xlsx", sheet="1"))
#defining inputs and outputs
inputs1=data.frame(df1[1:4])
outputs1=data.frame(df1[5:8])
```

```
inputs2=data.frame(df2[1:4])
outputs2=data.frame(df2[5:8])
inputs3=data.frame(df3[1:4])
outputs3=data.frame(df3[5:8])
inputs4=data.frame(df4[1:4])
outputs4=data.frame(df4[5:8])
n=dim(df1)[1]
m=dim(inputs1)[2]
s=dim(outputs1)[2]
#defining directions, objective, and right hand side
f.dir=c(rep("=",4*(m+s)), rep(">=",4*(N+m+s)+3))
for (j in 1:n)
{
f.obj= c(0,rep(0,m+s+n), 0,rep(0,m+s+n), 1,rep(0,m+s+n), 0, rep(0,m+s+n))
f.rhs=  c(rep(0,m),as.numeric(outputs1[j,]),  rep(0,m),as.numeric(outputs2[j,]),
rep(0,m),as.numeric(outputs3[j,]), as.numeric(inputs4[j,])*as.numeric(teta4[j,]),
as.numeric(outputs4[j,]), rep(0,4*(N+m+s)+2), as.numeric(-teta4[j,]))
#defining matrix of constraints' coefficient
con1.1=rbind(t(inputs1),t(outputs1))
con2.1=rbind(diag(m),matrix(0,s,m))
con3.1=rbind(matrix(0,m,s),-diag(s))
con4.1=rbind(t(-inputs1[j,]),matrix(0,s,1))
con1.2=rbind(t(inputs2),t(outputs2))
con2.2=rbind(diag(m),matrix(0,s,m))
con3.2=rbind(matrix(0,m,s),-diag(s))
con4.2=rbind(t(-inputs2[j,]),matrix(0,s,1))
con1.3=rbind(t(inputs3),t(outputs3))
con2.3=rbind(diag(m),matrix(0,s,m))
con3.3=rbind(matrix(0,m,s),-diag(s))
con4.3=rbind(t(-inputs3[j,]),matrix(0,s,1))
con1.4=rbind(t(inputs4),t(outputs4))
con2.4=rbind(diag(m),matrix(0,s,m))
con3.4=rbind(matrix(0,m,s),-diag(s))
con4.4=rbind(matrix(0,m,1),matrix(0,s,1))
f.con1=cbind(con4.1,con1.1,con2.1,con3.1)
f.con2=cbind(con4.2,con1.2,con2.2,con3.2)
f.con3=cbind(con4.3,con1.3,con2.3,con3.3)
f.con4=cbind(con4.4,con1.4,con2.4,con3.4)
f.con1.1=cbind(f.con1,matrix(0, (m+s), (3*(m+s+N+1))))
f.con2.1=cbind(matrix(0, (m+s), (m+s+N+1)), f.con2,matrix(0, (m+s),
(2*(m+s+N+1))))
```

```
f.con3.1 = cbind(matrix(0, (m+s), 2*(m+s+N+1)), f.con3,matrix(0, (m+s),
(m+s+N+1)))
f.con4.1=cbind(matrix(0, (m+s), 3*(m+s+N+1)), f.con4)
f.con.1 = rbind(f.con1.1, f.con2.1, f.con3.1, f.con4.1)
c.con1= cbind(matrix(0,N,1), matrix(1,N,1), matrix(0,N,m+s+N-1),
matrix(0,N,3*(m+s+N+1)))
c.con2= cbind(matrix(0,N,1), matrix(-1,N,1),matrix(0,N,m+s+N),matrix(1,N,1),
matrix(0,N,m+s+N-1),matrix(0,N,2*(m+s+n+1)))
c.con3= cbind(matrix(0,N,N+m+s+1), matrix(0,N,1), matrix(-1,N,1),matrix
(0,N,m+s+N),matrix(1,N,1),matrix(0,N,m+s+N-1),matrix(0,N,(m+s+n+1)))
c.con4= cbind(matrix(0,N,2*(N+m+s+1)), matrix(0,N,1),
matrix(-1,N,1),matrix(0,N,m+s+N),matrix(1,N,1),matrix(0,N,m+s+N-1))
f.con.2=rbind(c.con1,c.con2,c.con3,c.con4)
a.con1 = cbind(matrix(0,m,N+1), matrix(1,m,1),matrix(0,m,m+s-1),
matrix(0,m,3*(m+s+N+1)))
a.con2 = cbind(matrix(0,m,N+1), matrix(-1,m,1),matrix(0,m,m+s+N),
matrix(1,m,1), matrix(0,m,m+s-1),matrix(0,m,2*(m+s+N+1)))
a.con3 = cbind(matrix(0,m,(m+s+N+1)), matrix(0,m,N+1), matrix(-1,m,1),
matrix(0,m,m+s+N), matrix(1,m,1), matrix(0,m,m+s-1),matrix(0,m,(m+s+N
+1)))
a.con4 = cbind(matrix(0,m,2*(m+s+N+1)), matrix(0,m,N+1), matrix(-1,
m,1), matrix(0,m,m+s+N), matrix(1,m,1), matrix(0,m,m+s-1))
f.con.3=rbind(a.con1,a.con2,a.con3,a.con4)
b.con1 = cbind(matrix(0,s,N+1+m), matrix(1,s,1),matrix(0,s,s-1),
matrix(0,m,3*(m+s+N+1)))
b.con2 = cbind(matrix(0,s,N+1+m), matrix(-1,s,1),matrix(0,s,m+s+N),
matrix(1,s,1), matrix(0,s,s-1),matrix(0,m,2*(m+s+N+1)))
b.con3 = cbind(matrix(0,s,(m+s+N+1)), matrix(0,s,N+1+m), matrix(-1,s,1),
matrix(0,m,m+s+N), matrix(1,s,1), matrix(0,s,s-1),matrix(0,s,(m+s+N+1)))
b.con4 = cbind(matrix(0,m,2*(m+s+N+1)), matrix(0,s,N+1+m), matrix(-1,
s,1), matrix(0,s,m+s+N), matrix(1,s,1), matrix(0,s,s-1))
f.con.4=rbind(b.con1,b.con2,b.con3,b.con4)
d.con1= c(-1, rep(0,N+m+s), 1,rep(0,N+m+s), rep(0, 2*(N+m+s+1)))
d.con2= c(rep(0, (N+m+s+1)), -1, rep(0,N+m+s), 1,rep(0,N+m+s), rep(0,
(N+m+s+1)))
d.con3= c(rep(0, 2*(N+m+s+1)), -1, rep(0,N+m+s), 0,rep(0,N+m+s))
f.con.5= rbind(d.con1, d.con2, d.con3)
f.con= rbind(f.con.1, f.con.2, f.con.3, f.con.4, f.con.5)
#solving modell
results=lp ("min",f.obj, f.con, f.dir, f.rhs)
if (j==1)
{
```

```
effcrs=results$objval
}
else
{
effcrs=rbind(effcrs, results$objval)
}
}
ES = data.frame(effcrs)
colnames(EEs) = c('effi')
WriteXLS(EEs,"4.47.xls", row.names = F, col.names = F)
```

Assuming the optimal value of model (5.77) is teta3, the R code for model (5.78) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data71.xlsx", sheet="1"))
df2 = data.frame(read_excel(path = "data81.xlsx", sheet="1"))
df3 = data.frame(read_excel(path = "data81.xlsx", sheet="1"))
df4 = data.frame(read_excel(path = "data101.xlsx", sheet="1"))
teta4 = data.frame(read_excel(path = "teta4-(46).xlsx", sheet="1"))
teta3 = data.frame(read_excel(path = "teta3-(47).xlsx", sheet="1"))
#defining inputs and outputs
inputs1=data.frame(df1[1:4])
outputs1=data.frame(df1[5:8])
inputs2=data.frame(df2[1:4])
outputs2=data.frame(df2[5:8])
inputs3=data.frame(df3[1:4])
outputs3=data.frame(df3[5:8])
inputs4=data.frame(df4[1:4])
outputs4=data.frame(df4[5:8])
#number of elements
n=dim(df1)[1]
m=dim(inputs1)[2]
s=dim(outputs1)[2]
#defining directions, objective, and right hand side
f.dir=c(rep("=",4*(m+s)), rep(">=",4*(N+m+s)+3))
```

```
for (j in 1:n)
{
f.obj=c(0,rep(0,m+s+n), 1,rep(0,m+s+n), 0,rep(0,m+s+n), 0, rep(0,m+s+n))
f.rhs= c(rep(0,m),as.numeric(outputs1[j,]), rep(0,m),as.numeric(outputs2[j,]),
as.numeric(inputs3[j,])*as.numeric(teta3[j,]),as.numeric(outputs3[j,]),
as.numeric(inputs4[j,])*as.numeric(teta4[j,]),
as.numeric(outputs4[j,]),
rep(0,4*(N+m+s)+1),as.numeric(-teta3[j,]),0)
#defining matrix of constraints' coefficient
con1.1= rbind(t(inputs1),t(outputs1))
con2.1=rbind(diag(m),matrix(0,s,m))
con3.1=rbind(matrix(0,m,s),-diag(s))
con4.1=rbind(t(-inputs1[j,]),matrix(0,s,1))
con1.2 = rbind(t(inputs2),t(outputs2))
con2.2=rbind(diag(m),matrix(0,s,m))
con3.2=rbind(matrix(0,m,s),-diag(s))
con4.2=rbind(t(-inputs2[j,]),matrix(0,s,1))
con1.3 = rbind(t(inputs3),t(outputs3))
con2.3=rbind(diag(m),matrix(0,s,m))
con3.3=rbind(matrix(0,m,s),-diag(s))
con4.3=rbind(matrix(0,m,1),matrix(0,s,1))
con1.4 = rbind(t(inputs4),t(outputs4))
con2.4=rbind(diag(m),matrix(0,s,m))
con3.4=rbind(matrix(0,m,s),-diag(s))
con4.4=rbind(matrix(0,m,1),matrix(0,s,1))
f.con1 = cbind(con4.1,con1.1,con2.1,con3.1)
f.con2 = cbind(con4.2,con1.2,con2.2,con3.2)
f.con3 = cbind(con4.3,con1.3,con2.3,con3.3)
f.con4 = cbind(con4.4,con1.4,con2.4,con3.4)
f.con1.1= cbind(f.con1,matrix(0, (m+s), (3*(m+s+N+1))))
f.con2.1 = cbind(matrix(0, (m+s), (m+s+N+1)), f.con2,matrix(0, (m+s),
(2*(m+s+N+1))))
f.con3.1 <- cbind(matrix(0, (m+s), 2*(m+s+N+1)), f.con3,matrix(0, (m+s),
(m+s+N+1)))
f.con4.1 = cbind(matrix(0, (m+s), 3*(m+s+N+1)), f.con4)
f.con.1 = rbind(f.con1.1, f.con2.1, f.con3.1, f.con4.1)
c.con1= cbind(matrix(0,N,1), matrix(1,N,1), matrix(0,N,m+s+N-1), mtrix(0,
N,3*(m+s+N+1)))
c.con2= cbind(matrix(0,N,1),
matrix(-1,N,1),matrix(0,N,m+s+N),matrix(1,N,1),matrix(0,N,m+s
+N-1),matrix(0,N,2*(m+s+n+1)))
```

```
c.con3= cbind(matrix(0,N,N+m+s+1), matrix(0,N,1),
matrix(-1,N,1),matrix(0,N,m+s+N),matrix(1,N,1),matrix(0,N,m+s
+N-1),matrix(0,N,(m+s+n+1)))
c.con4= cbind(matrix(0,N,2*(N+m+s+1)), matrix(0,N,1),
matrix(-1,N,1),matrix(0,N,m+s+N),matrix(1,N,1),matrix(0,N,m+s+N-1))
f.con.2=rbind(c.con1,c.con2,c.con3,c.con4)
a.con1 = cbind(matrix(0,m,N+1), matrix(1,m,1),matrix(0,m,m+s-1),
matrix(0,m,3*(m+s+N+1)))
a.con2 = cbind(matrix(0,m,N+1), matrix(-1,m,1),matrix(0,m,m+s+N),
matrix(1,m,1), matrix(0,m,m+s-1),matrix(0,m,2*(m+s+N+1)))
a.con3 = cbind(matrix(0,m,(m+s+N+1)), matrix(0,m,N+1), matrix(-1,m,1),
matrix(0,m,m+s+N), matrix(1,m,1), matrix(0,m,m+s-1),matrix(0,m,(m+s+N
+1)))
a.con4 = cbind(matrix(0,m,2*(m+s+N+1)), matrix(0,m,N+1), matrix(-1,
m,1), matrix(0,m,m+s+N), matrix(1,m,1), matrix(0,m,m+s-1))
f.con.3=rbind(a.con1,a.con2,a.con3,a.con4)
b.con1 = cbind(matrix(0,s,N+1+m), matrix(1,s,1),matrix(0,s,s-1), matrix(0,
m,3*(m+s+N+1)))
b.con2 = cbind(matrix(0,s,N+1+m), matrix(-1,s,1),matrix(0,s,m+s+N), ma-
trix(1,s,1), matrix(0,s,s-1),matrix(0,m,2*(m+s+N+1)))
b.con3 = cbind(matrix(0,s,(m+s+N+1)), matrix(0,s,N+1+m), matrix(-1,s,1),
matrix(0,m,m+s+N), matrix(1,s,1), matrix(0,s,s-1),matrix(0,s,(m+s+N+1)))
b.con4 = cbind(matrix(0,m,2*(m+s+N+1)), matrix(0,s,N+1+m), matrix(-1,
s,1), matrix(0,s,m+s+N), matrix(1,s,1), matrix(0,s,s-1))
f.con.4=rbind(b.con1,b.con2,b.con3,b.con4)
d.con1= c(-1, rep(0,N+m+s), 1,rep(0,N+m+s), rep(0, 2*(N+m+s+1)))
d.con2= c(rep(0, (N+m+s+1)), -1, rep(0,N+m+s), 0,rep(0,N+m+s), rep(0,
(N+m+s+1)))
d.con3= c(rep(0, 2*(N+m+s+1)), 0, rep(0,N+m+s), 0,rep(0,N+m+s))
f.con.5= rbind(d.con1, d.con2, d.con3)
f.con= rbind(f.con.1, f.con.2, f.con.3, f.con.4, f.con.5)
#solving model
results = lp ("min",f.obj, f.con, f.dir, f.rhs)
if (j==1)
{
effcrs <- results$objval
}
else
{
effcrs <- rbind(effcrs, results$objval)
}
}
EES = data.frame(effcrs)
```

```
colnames(EES) = c('effi')
WriteXLS(EES,"4.48.xls", row.names = F, col.names = F)
```

Assuming the optimal value of model (5.78) is teta2, the R code for model (5.79) is as follows:

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data71.xlsx", sheet="1"))
df2 = data.frame(read_excel(path = "data81.xlsx", sheet="1"))
df3 = data.frame(read_excel(path = "data81.xlsx", sheet="1"))
df4 = data.frame(read_excel(path = "data101.xlsx", sheet="1"))
teta4 = data.frame(read_excel(path = "teta4-(46).xlsx", sheet="1"))
teta3 = data.frame(read_excel(path = "teta3-(47).xlsx", sheet="1"))
teta2 = data.frame(read_excel(path = "teta2-(48).xlsx", sheet="1"))
#defining inputs and outputs
inputs1=data.frame(df1[1:4])
outputs1=data.frame(df1[5:8])
inputs2=data.frame(df2[1:4])
outputs2=data.frame(df2[5:8])
inputs3=data.frame(df3[1:4])
outputs3=data.frame(df3[5:8])
inputs4=data.frame(df4[1:4])
outputs4=data.frame(df4[5:8])
#number of elements
n=dim(df1)[1]
m=dim(inputs1)[2]
s=dim(outputs1)[2]
#defining directions, objective, and right hand side
f.dir=c(rep("=",4*(m+s)), rep(">=",4*(N+m+s)+3))
for (j in 1:n)
{
=f.obj= c(0,rep(0,m+s+n), 1,rep(0,m+s+n), 0,rep(0,m+s+n), 0, rep(0,m+s
+n))
f.rhs= c(rep(0,m),as.numeric(outputs1[j,]), as.numeric(inputs2[j,])*as.nu-
meric(teta3[j,]),,as.numeric(outputs2[j,]),
as.numeric(inputs3[j,])*as.numeric(teta3[j,]),as.numeric(outputs3[j,]),
```

```
as.numeric(inputs4[j,])*as.numeric(teta4[j,]),
as.numeric(outputs4[j,]),
rep(0,4*(N+m+s)),as.numeric(-teta2[j,]),0,0)
#defining matrix of constraints' coefficient
con1.1=rbind(t(inputs1),t(outputs1))
con2.1=rbind(diag(m),matrix(0,s,m))
con3.1=rbind(matrix(0,m,s),-diag(s))
con4.1=bind(t(-inputs1[j,]),matrix(0,s,1))
con1.2=rbind(t(inputs2),t(outputs2))
con2.2=rbind(diag(m),matrix(0,s,m))
con3.2=rbind(matrix(0,m,s),-diag(s))
con4.2=rbind(matrix(0,m,1),matrix(0,s,1))
con1.3=rbind(t(inputs3),t(outputs3))
con2.3=rbind(diag(m),matrix(0,s,m))
con3.3=rbind(matrix(0,m,s),-diag(s))
con4.3=rbind(matrix(0,m,1),matrix(0,s,1))
con1.4=rbind(t(inputs4),t(outputs4))
con2.4=rbind(diag(m),matrix(0,s,m))
con3.4=rbind(matrix(0,m,s),-diag(s))
con4.4=rbind(matrix(0,m,1),matrix(0,s,1))
f.con1=cbind(con4.1,con1.1,con2.1,con3.1)
f.con2=cbind(con4.2,con1.2,con2.2,con3.2)
f.con3=cbind(con4.3,con1.3,con2.3,con3.3)
f.con4=cbind(con4.4,con1.4,con2.4,con3.4)
f.con1.1=cbind(f.con1,matrix(0, (m+s), (3*(m+s+N+1))))
f.con2.1=cbind(matrix(0, (m+s), (m+s+N+1)), f.con2,matrix(0, (m+s),
(2*(m+s+N+1))))
f.con3.1=cbind(matrix(0, (m+s), 2*(m+s+N+1)), f.con3,matrix(0, (m+s),
(m+s+N+1)))
f.con4.1=cbind(matrix(0, (m+s), 3*(m+s+N+1)), f.con4)
f.con.1 = rbind(f.con1.1, f.con2.1, f.con3.1, f.con4.1)
c.con1= cbind(matrix(0,N,1), matrix(1,N,1), matrix(0,N,m+s+N-1), matrix(0,
N,3*(m+s+N+1)))
c.con2= cbind(matrix(0,N,1), matrix(-1,N,1),matrix(0,N,m+s+N),matrix(1,
N,1),matrix(0,N,m+s+N-1),matrix(0,N,2*(m+s+n+1)))
c.con3= cbind(matrix(0,N,N+m+s+1), matrix(0,N,1), matrix(-1,N,1),matrix
(0,N,m+s+N),matrix(1,N,1),matrix(0,N,m+s+N-1),matrix(0,N,(m+s+n+1)))
c.con4= cbind(matrix(0,N,2*(N+m+s+1)), matrix(0,N,1), matrix(-1,N,1),
matrix(0,N,m+s+N),matrix(1,N,1),matrix(0,N,m+s+N-1))
f.con.2=rbind(c.con1,c.con2,c.con3,c.con4)
a.con1 = cbind(matrix(0,m,N+1), matrix(1,m,1),matrix(0,m,m+s-1), matrix
(0,m,3*(m+s+N+1)))
```

```
a.con2 = cbind(matrix(0,m,N+1), matrix(-1,m,1),matrix(0,m,m+s+N), matrix
(1,m,1), matrix(0,m,m+s-1),matrix(0,m,2*(m+s+N+1)))
a.con3 = cbind(matrix(0,m,(m+s+N+1)), matrix(0,m,N+1), matrix(-1,m,1),
matrix(0,m,m+s+N), matrix(1,m,1), matrix(0,m,m+s-1),matrix(0,m,(m+s+N
+1)))
a.con4 = cbind(matrix(0,m,2*(m+s+N+1)), matrix(0,m,N+1), matrix(-1,m,1),
matrix(0,m,m+s+N), matrix(1,m,1), matrix(0,m,m+s-1))
f.con.3=rbind(a.con1,a.con2,a.con3,a.con4)
b.con1 = cbind(matrix(0,s,N+1+m), matrix(1,s,1),matrix(0,s,s-1), matrix
(0,m,3*(m+s+N+1)))
b.con2 = cbind(matrix(0,s,N+1+m), matrix(-1,s,1),matrix(0,s,m+s+N),
matrix(1,s,1), matrix(0,s,s-1),matrix(0,m,2*(m+s+N+1)))
b.con3 = cbind(matrix(0,s,(m+s+N+1)), matrix(0,s,N+1+m), matrix(-1,s,1),
matrix(0,m,m+s+N), matrix(1,s,1), matrix(0,s,s-1),matrix(0,s,(m+s+N+1)))
b.con4 = cbind(matrix(0,m,2*(m+s+N+1)), matrix(0,s,N+1+m), matrix(-1,s,1),
matrix(0,s,m+s+N), matrix(1,s,1), matrix(0,s,s-1))
f.con.4=rbind(b.con1,b.con2,b.con3,b.con4)
d.con1= c(-1, rep(0,N+m+s), 0,rep(0,N+m+s), rep(0, 2*(N+m+s+1)))
d.con2= c(rep(0, (N+m+s+1)), 0, rep(0,N+m+s), 0,rep(0,N+m+s),
rep(0, (N+m+s+1)))
d.con3= c(rep(0, 2*(N+m+s+1)), 0, rep(0,N+m+s), 0,rep(0,N+m+s))
f.con.5= rbind(d.con1, d.con2, d.con3)
f.con= rbind(f.con.1, f.con.2, f.con.3, f.con.4, f.con.5)
#solving model
results=lp ("min",f.obj, f.con, f.dir, f.rhs)
if (j==1)
{
effcrs=results$objval
}
else
{
effcrs=rbind(effcrs, results$objval)
}
}
EEs = data.frame(effcrs)
colnames(EEs) = c('effi')
WriteXLS(EEs,"4.49.xls", row.names = F, col.names = F)
```

According the obtained optimal values of models (5.76)–(5.79), the R code for model (4.81) is as follows in which the optimal objective value of model (4.80) is teta1.

```
#required libraries
library(readxl)
library(lpSolve)
library(rJava)
library(WriteXLS)
#reading data
df1 = data.frame(read_excel(path = "data7.xlsx", sheet="1"))
df2 = data.frame(read_excel(path = "data8.xlsx", sheet="1"))
df3 = data.frame(read_excel(path = "data9.xlsx", sheet="1"))
df4 = data.frame(read_excel(path = "data10.xlsx", sheet="1"))
teta4 = data.frame(read_excel(path = "teta4-(46).xlsx", sheet="1"))
teta3 = data.frame(read_excel(path = "teta3-(47).xlsx", sheet="1"))
teta2 = data.frame(read_excel(path = "teta2-(48).xlsx", sheet="1"))
teta1 = data.frame(read_excel(path = "teta1-(49).xlsx", sheet="1"))
#defining inputs and outputs
inputs1=data.frame(df1[1:4])
outputs1=data.frame(df1[5:8])
inputs2=data.frame(df2[1:4])
outputs2=data.frame(df2[5:8])
inputs3=data.frame(df3[1:4])
outputs3=data.frame(df3[5:8])
inputs4=data.frame(df4[1:4])
outputs4=data.frame(df4[5:8])
#number of elements
n=dim(df1)[1]
m=dim(inputs1)[2]
s=dim(outputs1)[2]
#defining directions, objective, and right hand side
f.dir=c(rep("=",4*(m+s)), rep(">=",4*(N+m+s)))
for (j in 1:n)
{
f.obj= c(0,rep(0,n),rep(1,m+s), 0,rep(0,n),rep(1,m+s), 0,rep(0,n),rep(1,m+s),
0, rep(0,n),rep(1,m+s))
f.rhs=c(as.numeric(inputs1[j,])*as.numeric(teta1[j,]),as.numeric(outputs1
[j,]),
as.numeric(inputs2[j,])*as.numeric(teta2[j,]),as.numeric(outputs2[j,]),
as.numeric(inputs3[j,])*as.numeric(teta3[j,]),as.numeric(outputs3[j,]),
as.numeric(inputs4[j,])*as.numeric(teta4[j,]),as.numeric(outputs4[j,]),
rep(0,4*(N+m+s)))
#defining matrix of constraints' coefficient
con1.1=rbind(t(inputs1),t(outputs1))
con2.1=rbind(diag(m),matrix(0,s,m))
con3.1=rbind(matrix(0,m,s),-diag(s))
```

```
con4.1=rbind(matrix(0,m,1),matrix(0,s,1))
con1.2=rbind(t(inputs2),t(outputs2))
con2.2=rbind(diag(m),matrix(0,s,m))
con3.2=rbind(matrix(0,m,s),-diag(s))
con4.2=rbind(matrix(0,m,1),matrix(0,s,1))
con1.3=rbind(t(inputs3),t(outputs3))
con2.3=rbind(diag(m),matrix(0,s,m))
con3.3=rbind(matrix(0,m,s),-diag(s))
con4.3=rbind(matrix(0,m,1),matrix(0,s,1))
con1.4=rbind(t(inputs4),t(outputs4))
con2.4=rbind(diag(m),matrix(0,s,m))
con3.4=rbind(matrix(0,m,s),-diag(s))
con4.4=rbind(matrix(0,m,1),matrix(0,s,1))
f.con1=cbind(con4.1,con1.1,con2.1,con3.1)
f.con2=cbind(con4.2,con1.2,con2.2,con3.2)
f.con3=cbind(con4.3,con1.3,con2.3,con3.3)
f.con4=cbind(con4.4,con1.4,con2.4,con3.4)
f.con1.1=cbind(f.con1,matrix(0, (m+s), (3*(m+s+N+1))))
f.con2.1=cbind(matrix(0, (m+s), (m+s+N+1)), f.con2,matrix(0, (m+s),
(2*(m+s+N+1))))
f.con3.1=cbind(matrix(0, (m+s), 2*(m+s+N+1)), f.con3,matrix(0, (m+s),
(m+s+N+1)))
f.con4.1=cbind(matrix(0, (m+s), 3*(m+s+N+1)), f.con4)
f.con.1 = rbind(f.con1.1, f.con2.1, f.con3.1, f.con4.1)
c.con1= cbind(matrix(0,N,1), matrix(1,N,1), matrix(0,N,m+s+N-1), matrix
(0,N,3*(m+s+N+1)))
c.con2= cbind(matrix(0,N,1), matrix(-1,N,1),matrix(0,N,m+s+N),matrix
(1,N,1),matrix(0,N,m+s+N-1),matrix(0,N,2*(m+s+n+1)))
c.con3= cbind(matrix(0,N,N+m+s+1), matrix(0,N,1), matrix(-1,N,1),matrix
(0,N,m+s+N),matrix(1,N,1),matrix(0,N,m+s+N-1),
matrix(0,N,(m+s+n+1)))
c.con4= cbind(matrix(0,N,2*(N+m+s+1)), matrix(0,N,1), matrix(-1,N,1),matrix
(0,N,m+s+N),matrix(1,N,1),matrix(0,N,m+s+N-1))
f.con.2=rbind(c.con1,c.con2,c.con3,c.con4)
a.con1 = cbind(matrix(0,m,N+1), matrix(1,m,1),matrix(0,m,m+s-1), matrix
(0,m,3*(m+s+N+1)))
a.con2 = cbind(matrix(0,m,N+1), matrix(-1,m,1),matrix(0,m,m+s+N), matrix
(1,m,1), matrix(0,m,m+s-1),matrix(0,m,2*(m+s+N+1)))
a.con3 = cbind(matrix(0,m,(m+s+N+1)), matrix(0,m,N+1), matrix(-1,m,1),
matrix(0,m,m+s+N), matrix(1,m,1), matrix(0,m,m+s-1),
matrix(0,m,(m+s+N+1)))
a.con4 = cbind(matrix(0,m,2*(m+s+N+1)), matrix(0,m,N+1), matrix(-1,m,1),
matrix(0,m,m+s+N), matrix(1,m,1), matrix(0,m,m+s-1))
```

```
f.con.3=rbind(a.con1,a.con2,a.con3,a.con4)
b.con1 = cbind(matrix(0,s,N+1+m), matrix(1,s,1),matrix(0,s,s-1), matrix(0,s,3*
(m+s+N+1)))
b.con2 = cbind(matrix(0,s,N+1+m), matrix(-1,s,1),matrix(0,s,m+s+N),
matrix(1,s,1), matrix(0,s,s-1),matrix(0,s,2*(m+s+N+1)))
b.con3 = cbind(matrix(0,s,(m+s+N+1)), matrix(0,s,N+1+m), matrix(-1,s,1),
matrix(0,s,m+s+N), matrix(1,s,1), matrix(0,s,s-1),
matrix(0,s,(m+s+N+1)))
b.con4 = cbind(matrix(0,s,2*(m+s+N+1)), matrix(0,s,N+1+m), matrix(-1,s,1),
matrix(0,s,m+s+N), matrix(1,s,1), matrix(0,s,s-1))
f.con.4=rbind(b.con1,b.con2,b.con3,b.con4)
f.con= rbind(f.con.1, f.con.2, f.con.3, f.con.4)
#solving model
results=lp ("max",f.obj, f.con, f.dir, f.rhs)
if (j==1)
{
effcrs=results$objval
}
else
{
effcrs=rbind(effcrs, results$objval)
}
}
EEs = data.frame(effcrs)
colnames(EEs) = c('Sum')
WriteXLS(EEs,"4.50.xls", row.names = F, col.names = F)
```

Example 5.11 Consider the data given in Table 5.9. The results of solving models (5.76)–(5.80) are given in Table 5.14. The results indicate that DMUs A, B and C are fully FDEA efficient, DMU E is weak FDEA efiicient and the remaining DMUs are fully FDEA inefficient.

Table 5.14 Fuzzy efficiency of eight DMUs

DMU	Efficiency	Slack	Status
A	(1.00, 1.00, 1.00, 1.00)	0	Fully efficient
B	(1.00, 1.00, 1.00, 1.00)	0	Fully efficient
C	(1.00, 1.00, 1.00, 1.00)	0	Fully efficient
D	(1.00, 1.00, 1.00, 1.00)	0	Fully efficient
E	(1.00, 1.00, 1.00, 1.00)	2410.704	Weak inefficient
F	(0.93, 0.93, 0.93, 0.95)	0	Inefficient
G	(0.97, 0.97, 0.97, 0.97)	1611.82	Inefficient
H	(0.92, 0.92, 0.92, 0.92)	1868.246	Inefficient

5.6 Conclusion

In view of the fact that precise input and output data may not always be relevant in real world applications and are sometimes contains imprecise or vague data, several attempts have been made in the literature to deal with uncertain input and output data in DEA by use of fuzzy numbers. Therefore, in this chapter fuzzy DEA models have been reviewed from five various perspectives, namely alpha cut approach, fuzzy ranking approach, possibility approach, fuzzy arithmetic approach and MOLP approach to tackle fuzziness in input and output data. The resulting fuzzy DEA models have been formulated as linear programming models and presented by R codes for ease of solution and implementation. Illustrative examples have been provided to demonstrate the main advantages of R in FDEA models.

References

1. Emrouznejad, A., Tavana, M., Hatami-Marbini, A.: The state of the art in fuzzy data envelopment analysis. In: Performance Measurement with Fuzzy Data Envelopment Analysis. Studies in Fuzziness and Soft Computing, vol. 309, pp. 1–48. Springer, Berlin (2014)
2. Kao, C., Liu, S.T.: Fuzzy efficiency measures in data envelopment analysis. Fuzzy Sets Syst. **113**, 427–437 (2000)
3. Kao, C., Liu, S.T.: A mathematical programming approach to fuzzy efficiency ranking. Int. J. Prod. Econ. **86**, 145–154 (2003)
4. Saati, S., Memariani, A., Jahanshahloo, G.R.: Efficiency analysis and ranking of DMUs with fuzzy data. Fuzzy Optim. Decis. Making **1**, 255–267 (2002)
5. Saati, S., Memariani, A.: Reducing weight flexibility in fuzzy DEA. Appl. Math. Comput. **161**(2), 611–622 (2005)
6. Hatami-Marbini, A., Saati, S., Tavana, M.: An ideal-seeking fuzzy data envelopment analysis framework. Appl. Soft Comput. **10**, 1062–1070 (2010)
7. Kao, C., Liu, S.T.: Efficiencies of two-stage systems with fuzzy data. Fuzzy Sets Syst. **176**, 20–35 (2011)
8. Kao, C., Liu, S.T.: Efficiency of parallel production systems with fuzzy data. Fuzzy Sets Syst. **198**, 83–98 (2012)
9. Hatami-Marbini, A., Tavana, M., Emrouznejad, A., Saati, S.: Efficiency measurement in fuzzy additive data envelopment analysis. Int. J. Ind. Syst. Eng. **10**(1), 1–20 (2012)
10. Lozano, S.: Process efficiency of two-stage systems with fuzzy data. Fuzzy Sets Syst. **243**, 36–49 (2014)
11. Yager, R.R.: A characterization of the extension principle. Fuzzy Sets Syst. **18**, 205–217 (1986)
12. Zadeh, L.A., Outline of a new approach to the analysis of complex systems and decision processes, IEEE Trans. Systems Man Cybemet. SMC-1 (1973) 28-44
13. Zadeh, L.A.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets Syst. **1**, 3–28 (1978)
14. Zimmermann, H.J.: Fuzzy Set Theory and Its Applications, 2nd edn. Kluwer-Nijhoff, Boston (1991)
15. Chen, C.B., Klein, C.M.: A simple approach to ranking a group of aggregated fuzzy utilities, IEEE Trans. Syst. Man Cybernet. Part B: Cybemet. **27**, 26–35 (1997)
16. Chen, S.H.: Ranking fuzzy numbers with maximizing set and minimizing set. Fuzzy Sets Syst. **17**, 113–129 (1985)

17. Guo, P., Tanaka, H.: Fuzzy DEA: a perceptual evaluation method. Fuzzy Sets Syst. **119**, 149–160 (2001)
18. Leon, T., Liern, V., Ruiz, J.L., Sirvent, I.: A fuzzy mathematical programming approach to the assessment of efficiency with DEA models. Fuzzy Sets Syst. **139**, 407–419 (2003)
19. Soleimani-damaneh, M., Jahanshahloo, G.R., Abbasbandy, S.: Computational and theoretical pitfalls in some current performance measurement techniques and a new approach. Appl. Math. Comput. **181**, 1199–1207 (2006)
20. Guo, P., Tanaka, H.: Decision making based on fuzzy data envelopment analysis. In: Ruan, D., Meer, K. (eds.) Intelligent Decision and Policy Making Support Systems, pp. 39–54. Springer, Berlin (2008)
21. Guo, P.: Fuzzy data envelopment analysis and its application to location problems. Inf. Sci. **179**, 820–829 (2009)
22. Jahanshahloo, G.R., Hosseinzadeh Lotfi, F., Shahverdi, R., Adabitabar, M., Rostami-Malkhalifeh, M., Sohraiee, S.: Ranking DMUs by l_1-norm with fuzzy data in DEA. Chaos, Solitons Fractals **39**(5), 2294–2302 (2009)
23. Ebrahimnejad, A.: Cost efficiency measures with trapezoidal fuzzy numbers in data envelopment analysis based on ranking functions: application in insurance organization and hospital. Int. J. Fuzzy Syst. Appl. **2**(3), 51–68 (2012)
24. Ebrahimnejad, A.: A new link between output-oriented BCC model with fuzzy data in the present of undesirable outputs and MOLP. Fuzzy Inf. Eng. **2**, 113–125 (2011)
25. Ramik, J., Rimanek, J.: Inequality relation between fuzzy numbers and its use in fuzzy optimization. Fuzzy Sets Syst. **16**, 123–138 (1985)
26. Tanaka, H., Ichihasi, H., Asai, K.: A formulation of fuzzy linear programming problem based on comparison of fuzzy numbers. Control Cybern. **13**, 185–194 (1984)
27. Yao, J.-S., Wu, K.: Ranking fuzzy numbers based on decomposition principle and signed distance. Fuzzy Sets Syst. **116**, 275–288 (2000)
28. Lertworasirikul, S., Fang, S.C., Joines, J.A., Nuttle, H.L.W.: Fuzzy data envelopment analysis (DEA): a possibility approach. Fuzzy Sets Syst. **139**, 379–394 (2003)
29. Lertworasirikul, S., Fang, S.C., Nuttle, H.L.W., Joines, J.A.: Fuzzy BCC model for data envelopment analysis. Fuzzy Optim. Decis. Making **2**(4), 337–358 (2003)
30. Ruiz, J.L., Sirvent, I.: Fuzzy cross-efficiency evaluation: a possibility approach. Fuzzy Optim. Decis. Making **16**(1), 111–126 (2017)
31. Charnes, A., Cooper, W.W.: Chance-constrained programming. Manage. Sci. **6**, 73–79 (1959)
32. Wang, Y.M., Luo, Y., Liang, L.: Fuzzy data envelopment analysis based upon fuzzy arithmetic with an application to performance assessment of manufacturing enterprises. Expert Syst. Appl. **36**, 5205–5211 (2009)
33. Bhardwaj, B., Kaur, J., Kumar, A.: A new fuzzy CCR data envelopment analysis model and its application to manufacturing enterprises. In: Collan, M., Kacprzyk, J. (eds.) Soft Computing Applications for Group Decision-making and Consensus Modeling. Studies in Fuzziness and Soft Computing, vol. 357. Springer, Cham (2018)
34. Azar, A., Zarei Mahmoudabadi, M., Emrouznejad, A.: A new fuzzy additive model for determining the common set of weights in Data Envelopment Analysis. J. Intell. Fuzzy Syst. **30**(1), 61–69 (2016)
35. Hatami-Marbini, A., Ebrahimnejad, A., Lozano, S.: Fuzzy efficiency measures in data envelopment analysis using lexicographic multiobjective approach. Comput. Ind. Eng. **105**, 362–376 (2017)