Development of Data Imputation Methods for the Multiple Linear Regression

Thidarat Thongsri

A Thesis Submitted in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Applied Statistics

Prince of Songkla University

2022

Development of Data Imputation Methods for the Multiple Linear Regression

Thidarat Thongsri

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Applied Statistics

Prince of Songkla University

2022

Thesis Title Development of Data Imputation Methods for the Multiple Linear Regression

Author Miss Thidarat Thongsri

Major Program Applied Statistics

Major Advisor:

.....
(Asst. Prof. Dr. Klairung Samart)

Examining Committee:

.....Chairperson
(Assoc. Prof. Dr. Wararit Panichkitkosolkul)

.....Committee
(Asst. Prof. Dr. Klairung Samart)

.....Committee
(Asst. Prof. Dr. Phontita Thiuthad)

The Graduate School, Prince of Songkla University, has approved this thesis as partial fulfillment of the requirements for the Master of Science Degree in Applied Statistics.

.....
(Asst. Prof. Dr. Thakerng Wongsirichot)
Acting Dean of Graduate School

This is to certify that the work here submitted is the result of the candidate's own investigations. Due acknowledgement has been made of any assistance received.

.....Signature
(Asst. Prof. Dr. Klairung Samart)
Major Advisor

.....Signature
(Miss Thidart Thongsri)
Candidate

I hereby certify that this work has not been accepted in substance for any degree, and is not being currently submitted in candidature for any degree.

.....Signature
(Miss Thidarat Thongsri)
Candidate

ชื่อวิทยานิพนธ์ การพัฒนาวิธีการประมาณค่าสูญหายในตัวแบบการถดถอยเชิงเส้นพหุคูณ
ผู้เขียน นางสาวธิดารัตน์ ทองสี
สาขาวิชา สถิติประยุกต์
ปีการศึกษา 2565

บทคัดย่อ

การวิเคราะห์การถดถอยเชิงเส้นพหุคูณเป็นการวิเคราะห์ทางสถิติที่เกี่ยวข้องกับตัวแปรตามที่มีความสัมพันธ์กับตัวแปรอิสระมากกว่าหนึ่งตัว โดยการวิเคราะห์การถดถอยเชิงเส้นพหุคูณสามารถใช้ในการทำนายหรือประมาณค่าของตัวแปรตามได้ แต่ปัญหาสำคัญที่มักเกิดขึ้นเสมอในการวิเคราะห์ข้อมูลคือ การเกิดข้อมูลสูญหาย ซึ่งอาจจะทำให้ผลการวิเคราะห์ข้อมูลมีความคลาดเคลื่อนไปจากความเป็นจริงและสูญเสียรายละเอียดในบางส่วนที่สำคัญไป งานวิจัยนี้แบ่งออกเป็น 2 ส่วน ส่วนแรกมีวัตถุประสงค์เพื่อพัฒนาและเปรียบเทียบประสิทธิภาพของวิธีการประมาณค่าสูญหาย 8 วิธี ได้แก่ Hot deck imputation (HD), K-nearest neighbors imputation (KNN), Stochastic regression, imputation (SR), Predictive mean matching imputation (PMM), Random forest imputation (RF), Stochastic regression random forest with equivalent weight imputation (SREW), K-nearest random forest with equivalent weight imputation (KREW), และ K-nearest stochastic regression and random forest with equivalent weight imputation (KSREW) ในการศึกษานี้ใช้ตัวอย่างขนาด 30, 60, 100 และ 150 โดยมีเปอร์เซ็นต์การสูญหายที่ระดับ 10%, 20%, 30% และ 40% บนตัวแปรอิสระและตัวแปรตอบสนอง ใช้ Average mean square error (AMSE) ในการเปรียบเทียบประสิทธิภาพของวิธีการประมาณค่าสูญหาย ผลการวิจัยพบว่า การนำวิธีการประมาณค่าสูญหายมาผสมผสานกันมีประสิทธิภาพมากกว่าวิธีการประมาณค่าสูญหายแบบเดี่ยว และวิธี KSREW มีประสิทธิภาพในการประมาณค่าสูญหายดีที่สุด

งานวิจัยในส่วนที่ 2 มีวัตถุประสงค์เพื่อสร้างฟังก์ชันสำเร็จรูปในการวิเคราะห์การถดถอยเชิงเส้นพหุคูณแบบครบวงจร โดยใช้โปรแกรม RStudio ในชื่อของแพ็กเกจ *mlrpro* ซึ่งเป็นแพ็กเกจในการวิเคราะห์การถดถอยที่ใช้งานง่าย เหมาะสำหรับผู้เริ่มต้น เนื่องจากในตัวแพ็กเกจสามารถเลือกตัวแปรอิสระที่มีอิทธิพลต่อตัวแปรตาม สร้างตัวแบบการถดถอยที่ดีและเหมาะสมรวมถึงตรวจสอบข้อสมมุติเบื้องต้นของการวิเคราะห์การถดถอยและแปลงข้อมูลโดยใช้การแปลง Box-Cox แบบครบวงจร นอกจากนี้ในตัวแพ็กเกจ *mlrpro* สามารถคำนวณค่าสัมประสิทธิ์การถดถอย

ค่าส่วนเหลือ ค่าทำนายและค่าสถิติที่เกี่ยวข้องกับการวิเคราะห์การถดถอย อีกทั้งยังนำเสนอกราฟิกในรูปของกราฟต่าง ๆ ที่เกี่ยวข้องกับกับการวิเคราะห์การถดถอยเชิงเส้นพหุคูณ

Thesis Title Development of Data Imputation Methods for the Multiple Linear Regression
Author Miss Thidarat Thongsri
Major Program Applied Statistics
Academic Year 2022

ABSTRACT

Multiple linear regression is a statistical study that investigates the relationship between the response and the independent variables and may be used to predict or estimate the response values. Missing data is a serious issue that regularly occurs and impacts data analysis, resulting in the loss of information in certain critical areas and data analysis outcomes that differ greatly from reality.

This research is divided into two sections. The first project study's objective is to develop and compare the efficiency of eight imputation methods: hot deck imputation (HD), k-nearest neighbors imputation (KNN), stochastic regression imputation (SR), predictive mean matching imputation (PMM), random forest imputation (RF), stochastic regression random forest with equivalent weight imputation (SREW), k-nearest random forest with equivalent weight imputation (KREW), and k-nearest stochastic regression and random forest with equivalent weight imputation (KSREW). The simulation was done in this study with sample sizes of 30, 60, 100, and 150 with missing percentages of 10%, 20%, 30%, and 40% on both independent and response variables. The average mean square error (AMSE) was used to compare efficiency. The results reveal that the proposed composite approaches outperformed the single ones, particularly a three-component method called KSREW.

The second project is to create a function for analyzing multiple linear regressions using the RStudio software. The *mlrpro* package is an intuitive regression analysis tool that is suitable for novice users. It is a built-in package that can fit the regression model, select independent variables, validate the assumptions of multiple linear regression, transform data using the Box-Cox transformation, and determine which regression model is the most suited. The regression coefficients, residuals, fitted

values, and statistics related to regression, such as residual standard error, multiple R-squared, F-statistic, and so on, may all be obtained through the use of our *mlrpro* package. In addition to this, it provides visualization tools of the residuals plot, the normal Q-Q plot, and the lambda interval plot derived from Box-Cox transformations.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Asst. Prof. Dr. Klairung Samart, for her initial ideas, guidance, constant instruction, kindness and encouragement which enable me to carry out my study successfully.

My special appreciation is expressed to Assoc. Prof. Dr. Wararit Panichkitkosolkul and Asst. Prof. Dr. Phontita Thiuthad for valuable comments and helpful suggestions.

I wish to thank all my teachers in the Division of Computational Science, Prince of Songkla University for sharing their knowledge and support so that I can obtain this Master degree.

I greatfully appreciate my family for their love and encouragement. I would like to thank my friends for their encouragement and suggestions.

Finally, I would like to thank everyone, who supported me, but I did not mention above.

Thidarat Thonsgri

CONTENTS

Page

Abstract v
Acknowledgements ix
Contents x
List of tables xi
List of figures xii
List of papers 1
1. Introduction 3
1.1 Background 3
1.2 Objective 4
2. Literature reviews 4
3. Simulation study 7
4. Result and discussion 11
5. Conclusions 17
Bibliography 18
Appendices 22
Paper I 23
Paper II 34
Source code 47
Vitae 112

LIST OF TABLES

Tables **Page**

Table 4.1 AMSE of the eight methods when the standard deviation of the error term is 10 and the missing type is MCAR 11

Table 4.2 AMSE of the eight methods when the standard deviation of the error term is 15 and the missing type is MCAR 12

Table 4.3 AMSE of the eight methods when the standard deviation of the error term is 10 and the missing type is MAR 13

Table 4.4 AMSE of the eight methods when the standard deviation of the error term is 15 and the missing type is MAR 14

Table 4.5 AMSE of the eight methods when the standard deviation of the error term is 10 and the missing type is MNAR 15

Table 4.6 AMSE of the eight methods when the standard deviation of the error term is 15 and the missing type is MNAR 16

LIST OF FIGURES

Figures **Page**

Figure 3.1: Flowchart showing the comparison process of the eight imputation methods 7

Figure 3.2: Flowchart showing the process of the *mlrpro* package 8

LIST OF PAPERS

Paper I — Thongsri, T., & Samart, K. (2023). Development of Imputation Methods for Missing Data in Multiple Linear Regression Analysis. *Lobachevskii Journal of Mathematics*, 9(44).

Paper II — Thongsri, T., & Samart, K. mlrpro: Perform Stepwise Regression with Verifying Assumptions and Identifying Possible Box-Cox Transformation. *The R Journal. (Submitted).*

Reprints were made with permission from the publishers

Paper I

Lobachevskii Journal of Mathematics
ISSN: 1995-0802 (Print), 1818-9962 (Online)
Publisher: Pleiades Publishing, Ltd. Distributed by Springer.
Lobachevskii Institute of Mathematics and Mechanics,
Kazan Federal University, Kremlevskaya ul. 18,
Kazan, Tatarstan, 420008 Russia. Phone: +7 (843) 233-72-46;
e-mail: ljmeditor@gmail.com; http://ljm.kpfu.ru

10 July, 2022

Dear Professor Thidarat Thongsri and Professor Klairung Samart,

Ref: Development of Imputation Methods for Missing Data in Multiple Linear Regression Analysis

Our referees have now considered your paper and have recommended publication in Lobachevskii Journal of Mathematics. We are pleased to accept your paper in its current form which will now be forwarded to the publisher for copy editing and typesetting. Our plan is to publish your paper in Issue 9 (September) of Volume 44 (year 2023).

You will receive proofs for checking, and instructions for transfer of copyright in due course.

The publisher also requests that proofs are checked and returned within 48 hours of receipt.

Thank you for your contribution to Lobachevskii Journal of Mathematics and we look forward to receiving further submissions from you.

Sincerely,
Andrei Volodin
Editor, Lobachevskii Journal of Mathematics

1. Introduction

1.1 Background

Data analysis is frequently hampered by the significant problem of missing data, which can result in the loss of information in certain critical areas and lead the conclusions to diverge significantly from reality. Little and Rubin (2002) classify missing data into three mechanisms: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). In the literature, imputation of missing values and discarding missing values are the two methods for handling missing data (Han, Pei, & Kamber, 2011). The topic of this study has increased the number of choices that enhances imputation performance. As a result, we combined some methods namely, the k-nearest random forest with equivalent weighted imputation (KREW), a two-composite imputation method, as well as the k-nearest stochastic regression and random forest with equivalent weighted imputation (KSREW), a three-composite imputation method that combines the k-nearest neighbors, stochastic regression, and random forest methods and compared with the single ones. The missingness on both the response and independent variables came from three different types of mechanisms.

A variety of statistical applications are now available for analyzing multiple linear regressions. R programs is a tool for statistical data analysis that is available for free download and can be used to analyze multiple regression. The outputs of the program must be examined to determine whether independent variables influence response variable at the specified level of significance and to determine whether multiple regression analysis assumptions are met. If the assumptions are not met, the data must be investigated and transformed possibly using the Box-Cox transformation. Therefore, all users must have statistical understanding to accurately and dependably summarize and report the program's outcomes.

As is obvious, multiple linear regression analysis is quite complex and requires statistical understanding. So, the researchers desired to create an R package for complete multiple linear regression analysis.

1.2 Objective

1. To develop and compare different imputation methods for the multiple linear regressions, when independent and response variables are missed out with MCAR, MAR, and MNAR losses.
2. To create a function for analyzing multiple linear regressions using the RStudio software.

2. Literature reviews

The methods for imputation and mechanisms for missing data used in this study are described **In Paper I.**

In this study, the following imputation methods were studied and compared for efficacy: hot deck imputation (HD) is frequently used to impute non-response in surveys and this method is used by government statistics agencies and survey companies in numerous nations (Andridge & Little, 2010; Myers, 2011). The k-nearest neighbors imputation (KNN) is a popular imputation method that has received widespread application in the literature and it performs admirably in estimating missing values, when data is correlated (Hengpraprohm & Jungjit, 2018; Jadhav, Pramod, & Ramanathan, 2019; Lamjaisue, Thongteeraparp, & Sinsomboonthong, 2017; Pauzi, et al., 2021). The stochastic regression imputation (SR) and predictive mean matching imputation (PMM) can be efficiently estimated, when the sample size is large (Lamjaisue, Thongteeraparp, & Sinsomboonthong, 2017; Jadhav, Pramod, & Ramanathan, 2019). The random forest imputation (RF) is one of the most commonly used feature selection methods because of its ease of

use and it is effective when there is a moderate to high percentage of missing data (Tang & Ishwaran, 2017; Aguilera, Guardiola-Albert, & Serrano-Hidalgo, 2020).

According to studies of Lamjaisue, Thongteeraparp, & Sinsomboonthong (2017) and Thongsri & Samart (2022), utilizing equivalent weight to estimate missing data is more effective than using a single estimating approach. As a result, we proposed another two composite imputation methods: k-nearest random forest with equivalent weighted imputation (KREW), as well as a three-composite imputation method: k-nearest stochastic regression and random forest with equivalent weighted imputation (KSREW), which is a combination of k-nearest neighbors, stochastic regression, and random forest methods with equivalent weight. Then we compared them to six methods, namely; HD, KNN, PMM, SR, RF, and SREW when the missingness come from three types of mechanism (MCAR, MAR and MNAR) on both response and independent variables.

In Paper II, we studied the process for creating a multiple linear regression model, checking assumptions and Box-Cox transformation.

We viewed the multiple linear regression analysis. Calculation of regression coefficients, test for regression relation and creation of a regression model. Then, stepwise processes were studied. The stepwise regression method combines forward selection and backward deletion to be utilized in selecting independent variables that affect response variables (Smith, 2018; Walczak, 2000).

This package verifies the homoscedasticity and normality assumptions of the multiple linear regression model. Creating a plot of standardized residuals vs predicted values or testing with Levene's Test from the *car* package (car: Companion to Applied Regression) (Fox & Weisberg, 2018)is the fundamental method for determining whether homoscedasticity is satisfied. The multiple linear regression assumes that the residuals of the model have a normal distribution. We can examine the assumption visually with Q-Q plots or formally with statistical tests such as the Shapiro-Wilk and Kolmogorov-Smirnov tests (Royston J. P., 1982; Royston P. , 1995).

When the assumption fails, data transformation using the Box-cox transformation method is required. The Box-Cox transformation is a typical approach for transforming a nonnormally distributed data set into one that is more normally distributed. The goal of this method is to determine a value for lambda such that the transformed data is as close to normally distributed as possible without violating the normality or variance constancy of the error distributions (Box & Cox, 1964; Pengfei, 2005).

We use our understanding of the regression analysis procedure to develop a package in the R that simplifies and streamlines the entire approach for users.

3. Simulation study

This study consists of two parts. The first project, the simulation studies were conducted to compare the performance of the eight missing data imputation approaches. The eight techniques were evaluated using the average mean squared error (AMSE) and a description of its operation is shown in Figure 3.1.

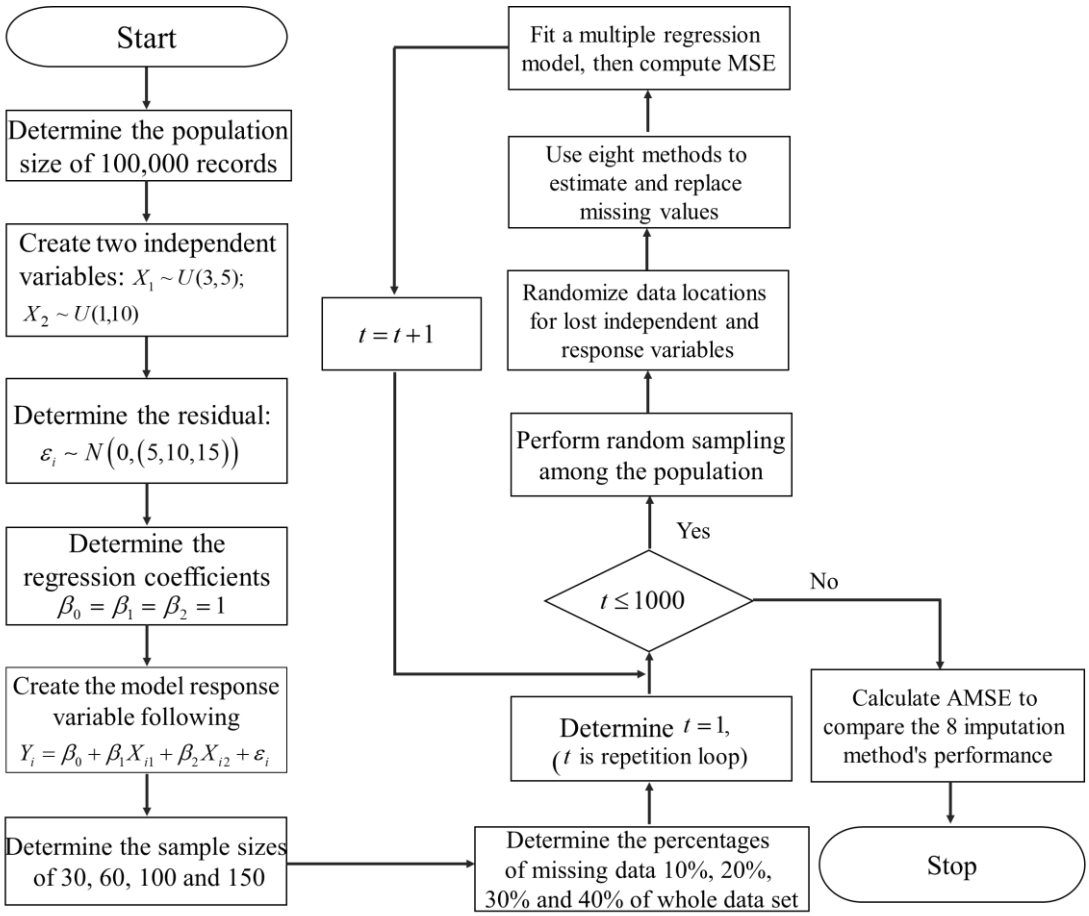

Figure 3.1: Flowchart showing the comparison process of the eight imputation methods.

The second project is to create an easy-to-use tool for analyzing multiple linear regression using the RStudio program. In the first stage, we will review the literature on all regression analyses. The package is then written and called *mlrpro* package. The flowchart and a description of its operation is shown in Figure 3.2.

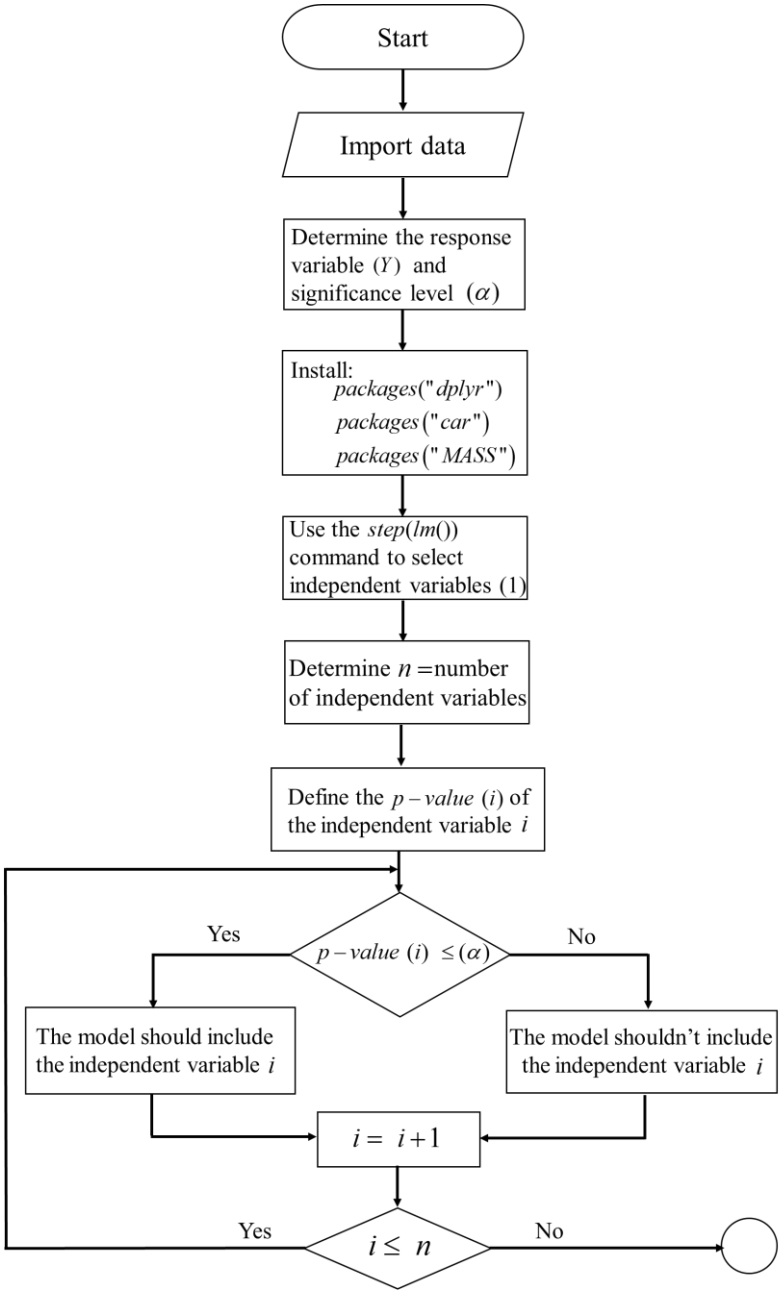

Figure 3.2: Flowchart showing the process of the *mlrpro* package**.**

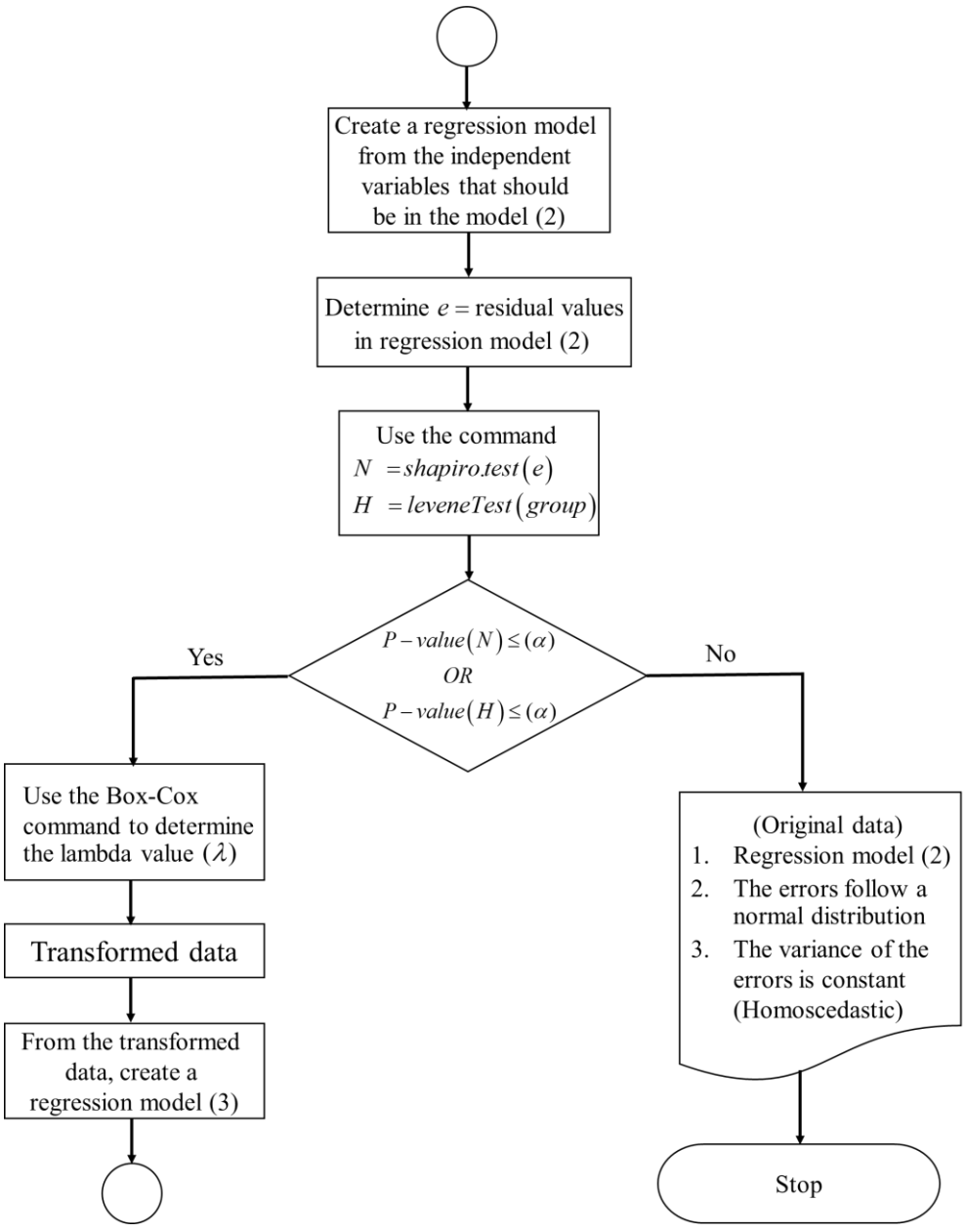

Figure 3.2: Flowchart showing the process of the *mlrpro* package.

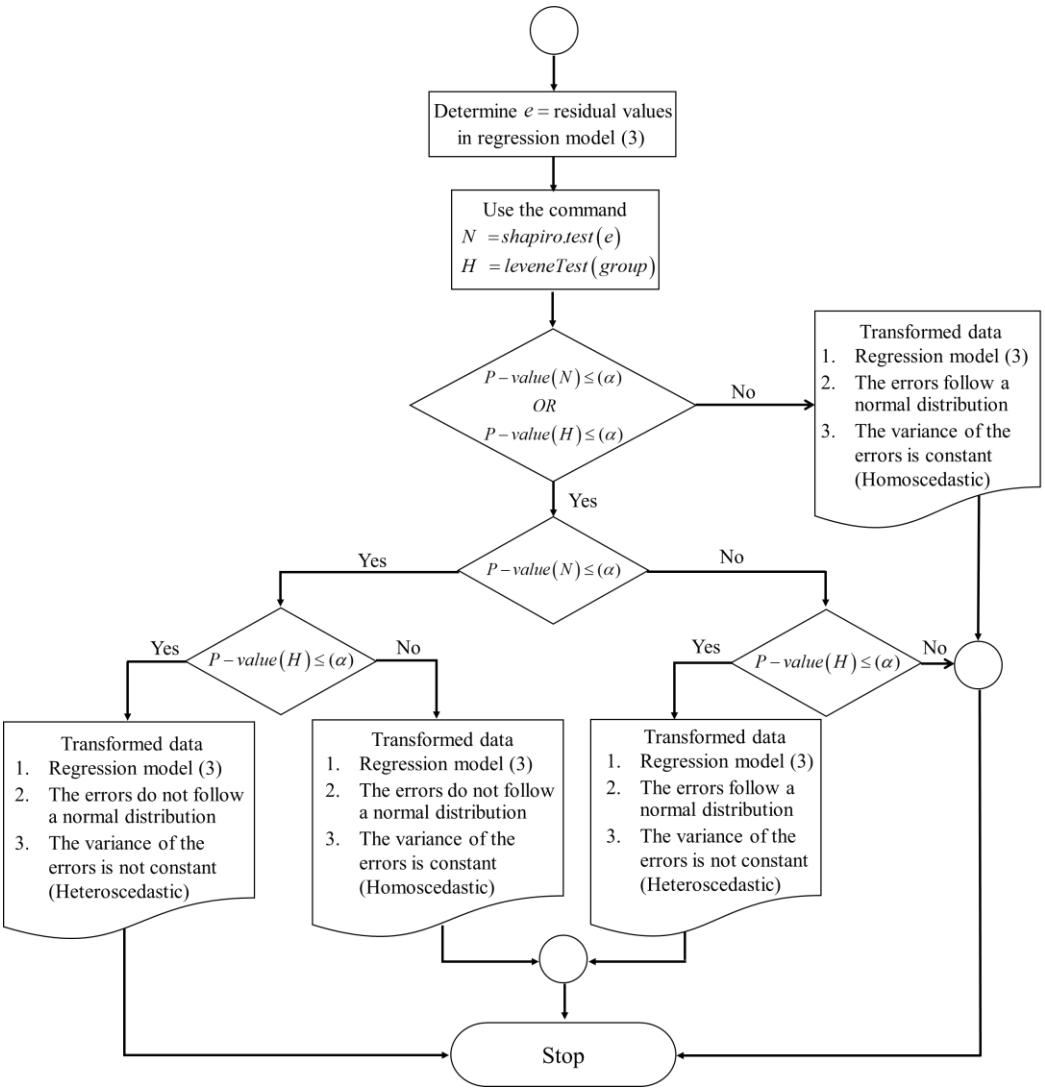

Figure 3.2: Flowchart showing the process of the *mlrpro* package.

4. Result and discussion

We present the simulation studies that were conducted to compare the performance of the eight missing data approaches **In Paper I**. The eight techniques were evaluated using the average mean squared error (AMSE).

The following results will not be shown **In Paper I**. They are extra research in case the standard deviations are between 10 and 15.

Table 4.1 AMSE of the eight methods when the standard deviation of the error term is 10 and the missing type is MCAR

Sample size	**Missing data (%)**	**Methods**							
		HD	**KNN**	**PMM**	**SR**	**RF**	**SREW**	**KREW**	**KSREW**
30	10	90.431	89.362	89.774	89.440	89.298	89.205	89.297	**89.201**
	20	91.729	89.585	90.486	89.852	89.550	89.342	89.482	**89.297**
	30	92.408	89.581	90.771	90.064	89.250	89.154	89.260	**89.085**
	40	93.800	89.370	91.245	90.123	89.184	88.949	89.102	**88.824**
60	10	95.978	95.041	95.355	95.216	95.030	95.003	95.013	**94.978**
	20	96.562	94.572	95.281	95.159	94.604	94.639	94.536	**94.534**
	30	96.227	93.296	94.572	94.058	93.370	93.355	93.251	**93.206**
	40	98.876	95.138	96.731	96.056	95.139	95.115	95.019	**94.934**
100	10	97.172	96.289	96.577	96.538	96.336	96.340	96.289	**96.283**
	20	98.507	96.923	97.497	97.452	96.989	97.034	96.902	**96.901**
	30	98.021	95.398	96.391	96.100	95.508	95.500	95.388	**95.361**
	40	99.377	96.237	97.616	97.341	96.374	96.429	**96.201**	96.207
150	10	98.922	**98.117**	98.362	98.351	98.158	98.168	98.122	98.125
	20	98.456	**96.859**	97.466	97.419	96.966	97.023	96.877	96.911
	30	100.034	**97.402**	98.450	98.260	97.652	97.679	97.468	97.491
	40	100.585	**97.127**	98.579	98.261	97.396	97.462	97.169	97.210

Table 4.2 AMSE of the eight methods when the standard deviation of the error term is 15 and the missing type is MCAR

Sample size	**Missing data (%)**	**Methods**							
		HD	**KNN**	**PMM**	**SR**	**RF**	**SREW**	**KREW**	**KSREW**
30	10	202.913	201.171	201.870	201.942	201.184	201.236	201.107	**201.107**
	20	203.600	199.841	201.829	200.186	199.636	199.254	199.594	**199.207**
	30	206.497	200.942	203.856	202.127	200.936	200.603	200.699	**200.343**
	40	206.332	199.348	202.824	201.505	199.190	199.052	198.929	**198.643**
60	10	213.841	212.475	213.163	212.802	212.535	212.493	212.467	**212.427**
	20	218.337	215.320	216.366	216.327	215.450	215.524	215.324	**215.296**
	30	219.012	214.587	216.526	216.236	214.920	214.977	214.549	**214.453**
	40	218.291	212.991	215.329	214.373	213.078	212.950	212.843	**212.674**
100	10	217.731	216.615	217.071	216.889	216.705	216.657	216.633	**216.599**
	20	218.992	216.621	217.558	217.192	216.776	216.692	216.640	**216.572**
	30	221.590	218.079	219.478	219.333	218.263	218.356	218.116	**218.078**
	40	221.745	216.904	219.428	218.842	217.175	217.419	217.040	**216.902**
150	10	220.551	**219.453**	219.878	219.780	219.562	219.551	219.488	219.484
	20	222.302	**220.026**	220.967	220.656	220.241	220.207	220.085	220.071
	30	223.142	**219.949**	221.481	221.093	220.175	220.283	219.985	220.051
	40	222.890	**218.282**	220.539	220.102	218.648	218.878	218.335	218.485

Table 4.3 AMSE of the eight methods when the standard deviation of the error term is 10 and the missing type is MAR

Sample size	**Missing data (%)**	**Methods**							
		HD	**KNN**	**PMM**	**SR**	**RF**	**SREW**	**KREW**	**KSREW**
30	10	89.786	88.857	89.134	89.016	88.744	88.727	88.765	**88.715**
	20	92.533	90.334	91.081	90.491	90.128	89.946	90.138	**89.930**
	30	92.350	89.540	90.429	89.506	89.148	**88.785**	89.195	88.824
	40	93.606	89.861	91.203	89.884	89.250	**88.819**	89.321	88.824
60	10	95.594	94.571	94.823	94.703	94.584	94.530	94.553	**94.506**
	20	96.979	95.228	95.639	95.413	95.143	95.055	95.128	**95.027**
	30	97.108	94.525	95.368	94.887	94.394	94.257	94.372	**94.202**
	40	97.564	94.493	95.516	94.761	94.028	93.906	94.110	**93.893**
100	10	97.483	96.576	96.847	96.762	96.580	96.579	96.559	**96.546**
	20	98.752	97.014	97.451	97.242	96.990	96.917	96.958	**96.881**
	30	98.686	96.445	96.991	96.775	96.366	96.263	96.328	**96.210**
	40	99.712	96.796	97.644	97.198	96.640	96.495	96.586	**96.416**
150	10	98.604	**97.750**	97.952	97.973	97.820	97.809	97.770	97.764
	20	99.476	97.908	98.304	98.189	97.923	97.872	97.875	**97.823**
	30	97.866	97.574	98.170	97.997	97.588	97.514	97.512	**97.430**
	40	100.948	98.155	98.936	98.555	97.986	97.892	97.950	**97.814**

Table 4.4 AMSE of the eight methods when the standard deviation of the error term is 15 and the missing type is MAR

Sample size	**Missing data (%)**	**Methods**							
		HD	**KNN**	**PMM**	**SR**	**RF**	**SREW**	**KREW**	**KSREW**
30	10	207.653	206.210	206.990	206.340	206.208	205.951	206.137	**205.925**
	20	203.980	200.146	201.370	200.850	200.157	199.865	199.990	**199.702**
	30	209.897	204.251	206.770	205.100	203.577	203.386	203.612	**203.249**
	40	210.306	203.396	206.800	205.020	202.891	202.689	202.768	**202.364**
60	10	215.561	214.026	214.480	214.230	214.115	213.982	214.033	**213.936**
	20	215.536	212.947	213.990	213.370	212.956	212.768	212.853	**212.681**
	30	217.825	214.126	215.740	215.140	214.016	214.016	213.925	**213.829**
	40	220.720	216.106	217.700	216.770	215.634	215.428	215.621	**215.333**
100	10	217.293	216.014	216.570	216.260	216.074	216.032	216.016	**215.984**
	20	221.908	219.800	220.570	220.170	219.692	219.660	219.685	**219.611**
	30	219.676	216.491	217.520	216.850	216.333	216.146	216.303	**216.101**
	40	221.795	218.021	219.030	218.420	217.371	217.339	217.498	**217.308**
150	10	221.394	**220.317**	220.660	220.560	220.409	220.375	220.344	220.323
	20	222.428	220.604	221.170	221.080	220.629	220.623	220.568	**220.539**
	30	221.590	218.842	219.610	219.320	218.807	218.701	218.730	**218.610**
	40	223.131	219.786	220.750	220.100	219.551	219.322	219.501	**219.247**

Table 4.5 AMSE of the eight methods when the standard deviation of the error term is 10 and the missing type is MNAR

Sample size	**Missing data (%)**	**Methods**							
		HD	**KNN**	**PMM**	**SR**	**RF**	**SREW**	**KREW**	**KSREW**
30	10	91.625	90.554	91.069	90.891	90.515	90.551	90.496	**90.493**
	20	92.779	90.861	91.680	91.122	90.690	90.545	90.691	**90.520**
	30	94.091	90.729	92.124	90.816	90.360	**90.052**	90.417	90.079
	40	96.677	92.723	94.350	93.025	92.349	91.924	92.342	**91.879**
60	10	95.883	94.924	95.285	95.081	94.923	94.889	94.899	**94.865**
	20	97.322	95.609	96.235	95.920	95.579	95.513	95.540	**95.463**
	30	97.501	94.908	96.019	95.436	94.956	94.826	94.852	**94.723**
	40	99.512	95.780	97.292	96.509	95.437	95.478	95.477	**95.378**
100	10	98.189	97.291	97.605	97.487	97.314	97.306	97.284	**97.270**
	20	98.110	96.448	97.166	96.839	96.513	96.476	96.437	**96.397**
	30	99.716	97.211	98.159	97.789	97.267	97.221	97.169	**97.112**
	40	99.591	96.366	97.569	97.103	96.272	96.260	96.214	**96.133**
150	10	98.406	97.602	97.873	97.807	97.647	97.640	97.609	**97.602**
	20	99.916	98.257	98.784	98.657	98.326	98.310	98.256	**98.234**
	30	100.720	98.249	99.098	98.940	98.290	98.345	**98.207**	98.218
	40	101.028	97.720	98.843	98.4510	97.724	97.716	97.618	**97.565**

Table 4.6 AMSE of the eight methods when the standard deviation of the error term is 15 and the missing type is MNAR

Sample size	Missing data (%)	Methods							
		HD	**KNN**	**PMM**	**SR**	**RF**	**SREW**	**KREW**	**KSREW**
30	10	205.319	203.715	204.540	203.770	203.558	**203.339**	203.576	203.358
	20	203.705	199.972	201.590	201.230	199.878	199.928	199.781	**199.717**
	30	208.540	202.809	205.070	203.630	202.544	202.031	202.436	**201.902**
	40	212.187	205.117	208.060	206.610	204.221	204.088	204.347	**203.894**
60	10	215.670	214.271	214.980	214.680	214.345	214.325	214.269	**214.247**
	20	216.554	213.878	215.050	214.720	213.871	219.933	213.800	**213.784**
	30	218.031	214.315	216.180	215.450	214.356	214.316	214.210	**214.109**
	40	221.539	216.290	218.520	217.800	215.964	216.073	215.928	**215.841**
100	10	219.404	218.356	218.820	218.690	218.394	218.404	218.349	**218.346**
	20	220.320	217.817	218.840	218.360	217.866	217.822	217.781	**217.722**
	30	223.237	219.654	221.110	220.710	219.769	219.777	219.611	**219.575**
	40	221.366	216.952	218.790	217.850	216.765	216.715	216.713	**216.576**
150	10	221.666	**220.568**	221.000	220.860	220.670	220.650	220.599	220.591
	20	223.313	221.282	222.050	221.780	221.391	221.344	221.290	**221.249**
	30	223.116	220.046	221.340	221.090	220.070	220.218	**219.982**	220.036
	40	224.879	220.907	222.270	221.710	220.763	220.737	220.701	**220.599**

Tables 4.1-4.6 reveal that when the standard deviation of error increased to 10 and 15, the findings matched the simulation with a standard deviation of error of 5. Although increasing the sample size to 200, 300, and 500, the outcome stays the same.

5. Conclusions

The first project study's objective is to develop and compare the efficiency of eight methods for dealing with missing data on dependent and independent variables in multiple linear regression. The estimates obtained by the composite imputation method: SREW, KREW, and KSREW outperform other approaches, according to our simulation results. The KSREW, in particular, excelled in practically every situation. Furthermore, when the sample size was less than 60, the SREW performed well next to the KSREW, but when the sample size was greater than 60, the KREW seemed to perform better. Overall, the proposed composite approaches outperformed the single ones, particularly a three-component method called KSREW.

The second project is to create a function for analyzing multiple linear regressions using the RStudio software. The *mlrpro* package is a tool for multiple regression, select independent variables, check multiple linear regression assumptions and identify possible model. We have developed functions to calculate the best multiple linear regression model, coefficients, fitted values and residuals etc. Moreover, it can compute lambda value, transform data by Box-Cox transformation and present graphical displays of residuals versus fitted values plot, normal Q-Q plot and lambda interval plot derived from Box-Cox transformations. The *mlrpro* package is ideal for users who are just starting out with regression analysis because it is simple and straightforward for users. Additionally, we expect that the accessibility of this tool will lead to an increase in their overall utilization.

Bibliography

[1] Aguilera, H., Guardiola-Albert, C., & Serrano-Hidalgo, C. (2020). Estimating extremely large amounts of missing precipitation data. *Journal of Hydroinformatics, 22*(3), 578-592.

[2] Alboukadel, K. (2019). *datarium: Data Bank for Statistical Analysis and Visualization*. Retrieved from https://CRAN.R-project.org/package=datarium.

[3] Andridge, R. R., & Little, R. J. (2010). A review of hot deck imputation for survey non-response. *International statistical review, 78*(1), 40-64.

[4] Atkinson, A. C. (1985). *Plots, Transformations and Regression.* Oxford University Press.

[5] Best, K. B., Gilligan, J. M., Baroud, H., Carrico, A. R., Donato, K. M., Ackerly, B. A., & Mallick, B. (2017). Random forest analysis of two household surveys can identify important predictors of migration in Bangladesh. *Journal of Computational Social Science, 4*(1), 77-100.

[6] Box, G. E., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological), 26*(2), 211-243.

[7] Chambers, J. M. (1992). *Linear models. Chapter 4 of statistical models in S.* Wadsworth & Brooks/Cole.

[8] Chambers, J. M., & Hastie, T. J. (1992). *Linear models. Chapter 4 of statistical models in S.* Wadsworth & Brooks/Cole.

[9] Chambers, J. M., Cleveland, W. S., Kleiner, B., & Tukey, P. A. (1983). *Graphical methods for data analysis.* Belmont, CA: Wadsworth.

[10] Chaovanaphan, P., & Chaimongkol., W. (2017). A comparison of the estimation methods for missing data in sample survey. *The Journal of Applied Science, 16*(1), 60-73.

[11] Cortez, P., & Morais, A. D. (2007). *A data mining approach to predict forest fires using meteorological data.*

[12] Enders, C. (2010). *Applied missing data analysis.* Guilford press.

[13] Faraway, J. J. (2004). *Linear models with R.* Chapman and Hall/CRC.

[14] Fox, J. (2015). *Applied regression analysis and generalized linear models.* Sage Publications.

[15] Fox, J., & Weisberg, S. (2018). *An R companion to applied regression.* Sage publications.

[16] Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques.* Elsevier.

[17] Hengpraprohm, K., & Jungjit, S. (2018). Missing Value Imputation Method using Ensemble Technique For Microarray Data. *Information Technology Journal, 14*(2), 9-17.

[18] Hong, S., & Lynn, H. S. (2020). Accuracy of random-forest-based imputation of missing data in the presence of non-normality, non-linearity, and interaction. *medical research methodology, 20*(1), 1-12.

[19] Hülya, Ö. Z., & Cengiz, B. A. (2020). A study on missing data problem in random Forest. *Osmangazi Tip Dergisi, 42*(1), 103-109.

[20] Ihaka, R., & Gentleman, R. (1996). R: a language for data analysis and graphics. *Journal of computational and graphical statistics, 5*(3), 299-314.

[21] Jadhav, A., Pramod, D., & Ramanathan, K. (2019). Comparison of performance of data imputation methods for numeric dataset. *Applied Artificial Intelligence, 33*(10), 913-933.

[22] Jim., W. (2019). *The Advantages of Regression Analysis & Forecasting*. Retrieved from https://bizfluent.com/info-12089071-advantages-regression-analysis-forecasting.html.

[23] Lamjaisue, R., Thongteeraparp, A., & Sinsomboonthong, J. (2017). Comparison of missing data estimation methods for the multiple regression analysis with missing at random dependent variable. *Science and Technology Journal, 25*(5), 766-777.

[24] Little, R. J. (1988). Missing-data adjustments in large surveys. *Journal of Business & Economic Statistics, 6*(3), 287-296.

[25] Little, R. J., & Rubin, D. B. (2002). *Statistical analysis with missing data.* John Wiley & Sons.

[26] Muliwan, P., Chutiman, N., & Pue-on, P. (2014). Development of Hot-deck Corrected Item Mean (HDD-CIM) for Estimating Missing Data. *Journal of Science and Technology Mahasarakham University, 33*(2), 175-178.

[27] Myers, T. A. (2011). Goodbye, listwise deletion: Presenting hot deck imputation as an easy and effective tool for handling missing data. *Communication methods and measures, 5*(4), 297-310.

[28] Neter, J., Kutner, M. H., Nachtsheim, C. J., & Wasserman, W. (1996). *Applied linear statistical models.*

[29] Pauzi, M., Azifah, N., Wah, Y. B., Deni, S. M., Rahim, N. A., & Khatijah, S. (2021). Comparison of Single and MICE Imputation Methods for Missing Values: A Simulation Study. *Pertanika Journal of Science & Technology, 29*(2), 979-998.

[30] Pengfei, L. (2005). *Box-Cox transformations: an overview*. Retrieved from https://www.ime.usp.br/~abe/lista/pdfm9cJKUmFZp.pdf.

[31] Royston, J. P. (1982). Algorithm AS 181: the W test for normality. *Applied Statistics*, 176-180.

[32] Royston, J. P. (1982). An extension of Shapiro and Wilk's W test for normality to large samples. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *31*(2), 115-124.

[33] Royston, P. (1995). Remark AS R94: A remark on algorithm AS 181: The W-test for normality. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *44*(4), 547-551.

[34] Rubin, D. B. (1986). Statistical matching using file concatenation with adjusted weights and multiple imputations. *Journal of Business & Economic Statistics, 4*(1), 87-94.

[35] Samart, K., Jansakul, N., & Chongcheawchamnan, M. (2018). Exact bootstrap confidence intervals for regression coefficients in small samples. *Communications in Statistics-Simulation and Computation, 47*(10), 2953-2959.

[36] Samart, K., Jansakul, N., & Chongcheawchamnan, M. (2018). Exact bootstrap confidence intervals for regression coefficients in small samples. *Communications in Statistics-Simulation and Computation*, 2953-2959.

[37] Schafer, J. L., & Graham, J. W. (2002). Missing data: our view of the state of the art. *Psychological methods, 7*(2), 147-177.

[38] Smith, G. (2018). Step away from stepwise. *Journal of Big Data, 5*(1), 1-12.

[39] Stekhoven, D. J., & Bühlmann, P. (2012). MissForest-non-parametric missing value imputation for mixed-type data. *Bioinformatics, 28*(1), 112-118.

[40] Tang, F., & Ishwaran, H. (2017). Random forest missing data algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal, 10*(6), 363-377.

[41] Thinh, R., Samart, K., & Jansakul, N. (2020). Linear regression models for heteroscedastic and non-normal data. *SCIENCEASIA, 46*(3), 353-360.

[42] Thongsri, T., & Samart, K. (2022). Composite Imputation Method for the Multiple Linear Regression with Missing at Random Data. *International Journal of Mathematics and Computer Science, 17*(1), 51-62.

[43] Thongsri, T., & Samart, K. (2022). *mlrpro: Perform Stepwise Regression with Verifying Assumptions and Identifying Possible Box-Cox Transformation*. Retrieved from https://CRAN.Rproject.org/package=mlrpro.

[44] Van Buuren, S. (2018). *Flexible imputation of missing data.* CRC press.

[45] Venables, W. N., & Ripley, B. D. (2002). *Modern Applied Statistics with S.* New York: Springer (4th ed).

[46] Vink, G., Frank, L. E., Pannekoek, J., & Van Buuren, S. (2014). Predictive mean matching imputation of semicontinuous variables. *Statistica Neerlandica, 61*(8), 61-90.

[47] Walczak, B. (2000). *Wavelets in chemistry.* Elsevier.

[48] Wilkinson, G. N., & Rogers, C. E. (1973). Symbolic description of factorial models for analysis of variance. *Journal of the Royal Statistical Society: Series C (Applied Statistics), 22*(3), 392-399.

[49] Wongarmart, A. (2012). *Comparison of the estimation methods for nonignorable missing data in multiple linear regression.* M.S. Dissertation. Thailand: Chulalongkorn University.

APPENDICES

Paper I

Development of Imputation Methods for Missing Data in Multiple Linear Regression Analysis

Thongsri T., and Samart K.

Development of Imputation Methods for Missing Data in Multiple Linear Regression Analysis

Thidarat Thongsri[1] and Klairung Samart[1,*]

(Submitted by A. I. Volodin)

[1] *Statistics and Applications Research Unit, Division of Computational Science, Faculty of Science, Prince of Songkla University, Songkhla, Thailand*

Received July 11, 2022; revised July 23, 2022; accepted August 15, 2022

Abstract—Missing data is a common issue in many domains of study. If this issue is disregarded, the erroneous conclusion may be reached. This study's objective is to develop and compared the efficiency of eight imputation methods: hot deck imputation (HD), k-nearest neighbors imputation (KNN), stochastic regression, imputation (SR), predictive mean matching imputation (PMM), random forest imputation (RF), stochastic regression random forest with equivalent weight imputation (SREW), k-nearest random forest with equivalent weight imputation (KREW), and k-nearest stochastic regression and random forest with equivalent weight imputation (KSREW). In this study, the simulation was run using sample sizes of 30, 60, 100, and 150, and missing percentages of 10%, 20%, 30%, and 40%. The average mean square error (AMSE) was used to compare efficiency. The results reveal that the proposed composite approaches outperformed the single ones, particularly a three-component method called KSREW. Increasing the number of components to a four-component method, on the other hand, has no effect on imputation performance.

2010 Mathematical Subject Classification: 12345, 54321

Keywords and phrases: *Missing data, imputation method, composite method, multiple linear regression*

1. INTRODUCTION

Multiple linear regression analysis is a statistical approach for examining the relationship between the response and the independent variables [19, 23]. Missing data is a common concern during data collection. This is an important issue to consider because disregarding it might lead to untrustworthy results. Different techniques are required for different types of missing data.

Little and Rubin [14] classify missing data into three mechanisms: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). A missing value is classified as MCAR if it is random and independent to any other variable. A missing value is considered as MAR if it is not random and is dependent on observed values in the dataset. If a missing value is dependent on both observed and unobserved values, it is categorized as MNAR [24]. These various types of missing data have a substantial impact on how statistically significant missing data should be treated.

Several strategies for dealing with missing data have been proposed in the literature. There are two ways for dealing with missing data disregarding missing values and imputation of missing values [7]. The first method simply rejects cases with missing data. It is commonly used in general because it is the easiest way for dealing with missing data without extensive statistical understanding [11]. The downside of this strategy, however, is that the sample size diminishes, resulting in biased estimates and loss of precision [20]. As a result, this strategy is suitable for datasets with a low proportion of missing values [24].

* E-mail: klairung.s@psu.ac.th

Imputation of missing values is a more appealing method. This method is simply an estimation of reasonable values to fill in the blanks. The imputation approach has the advantage of reducing bias owing to missingness rather than eliminating incomplete cases [11]. Several imputation methods have been established by researchers such as hot deck (HD), stochastic regression (SR), k-nearest neighbors (KNN), predictive mean matching (PMM) and random forest (RF) [9, 11, 13, 14, 18].

Imputation procedures have been thoroughly researched in a number of research articles. For example, Jadhav, Pramod and Ramanathan [11] employed five different numeric datasets to examine seven methods: median imputation, mean imputation, KNN, PMM, Bayesian linear regression, non-Bayesian linear regression and random sample. The KNN outperforms the other approaches. The performance of the data imputation method has also been discovered to be independent of the dataset and the percentage of missing values in the dataset. Hong and Lynn [9] compared three methods for handling missing data with missing at random (MAR): missForest CALIBERrfimpute and PMM. Both CALIBERrfimpute and missForest have a good prediction accuracy. The missForest can give regression coefficient estimates and confidence interval coverages, especially in nonlinear models. When estimating regression coefficients in linear models, the CALIBERrfimpute often outperforms missForest, while the PMM is used for logistic regression. Hulya and Cengiz [10] compared the proximity matrix imputation approach based on random forest and the KNN in a variety of cases, including varied percentages of missing data, number of neighbors (k), and correlation structures between predictor variables. The results showed that proximity matrix should be utilized for highly correlated structures, whereas the KNN should be used for low and medium correlated structures. Thongsri and Samart [24] compared six methods for handling missing data in multiple linear regression at random on both response and independent variables: listwise deletion (LD), HD, PMM, SR, RF, and stochastic regression random forest with equivalent weight (SREW). The findings indicate that the SREW is the most efficient in all cases.

The subject addressed in this study, which builds on earlier research, is whether increasing the number of alternatives improves imputation performance. As a result, we proposed another two-composite imputation method: k-nearest random forest with equivalent weighted imputation (KREW), as well as a three-composite imputation method: k-nearest stochastic regression and random forest with equivalent weighted imputation (KSREW), which is a combination of k-nearest neighbors, stochastic regression, and random forest methods with equivalent weight. Then we compared them to six methods, namely; HD, KNN, PMM, SR, RF, and SREW when the missingness come from three types of mechanism (MCAR, MAR and MNAR) on both response and independent variables.

The following is how this article is structured. Section 2 introduces the two composite imputation methods as well as the six imputation methods employed in this work. Section 3 then delves into the performance of the imputation approaches through simulated studies in various settings. Section 4 illustrates the outcomes of the application to real-world data. Section 5 contains the conclusions, including a discussion of the findings.

2. IMPUTATION METHODS

Let Y_i be a random variable and $X_{i1}, X_{i2}, X_{i3}, ..., X_{iq}$ be independent variables, where $i = 1, 2, 3, ..., n$. The methods for imputation missing data used in this study are as follows.

2.1. Hot deck imputation

Hot deck imputation (HD) is frequently used to impute non-response in surveys [2]. It is a method for dealing with missing data in which each missing value (the recipient) is replaced by an observation of the response of a similar unit (the donor) [2, 11, 15]. The hot deck method comes in two varieties. The random hot deck method is used when the donor is chosen at random from a pool of potential donors. The deterministic hot deck method is used when a single donor is identified based on some metric and values are imputed from that case [6].

The hot deck has appealing elements, such as it does not rely on model fitting to impute the variable. Furthermore, compared to a parametric model based on the imputation method, it might be less prone to model misspecification. Because they are derived from observed donor pool responses, another benefit is that only reasonable values are imputed [11]. As a result, this method is used by government statistics agencies and survey companies in numerous nations [16].

2.2. K-nearest neighbors imputation

K-nearest neighbors imputation (KNN) is a popular imputation method that has received widespread application in the literature [8, 11, 12, 17]. The KNN is a non-parametric method, which is usually used for classification and regression problems. Furthermore, the KNN may be used to replace missing values with values from the nearest neighbors.

This method averages values from related entries in the same dataset to impute missing values. A prediction model does not have to be made for each attribute. The drawback of these methods is that it takes a while to analyze big datasets [11].

The number of k neighbors is determined by a distance metric that varies based on the type of data, such as Hamming distance for categorical data Euclidean distance for continuous data, and different k numbers are likely to yield different results. As a result, selecting an appropriate value k is crucial [11, 12].

The KNN method can be divided into 3 steps as follows Lamjaisue, Thongteeraparp and Sinsomboonthong [12]:

1. Define k, where $k = \sqrt{m}$ and m is the amount of complete data.
2. Calculate the distance between the missing rows and the complete rows using the euclidean distance method $dist(R_i, R_j) = \sqrt{\sum_{q=1}^{p}(X_{iq} - X_{jq})^2}$, where $dist(R_i, R_j)$ denotes the distance between the missing rows and the complete rows. X_{iq} denotes missing data at row i and column q. X_{jq} denotes complete data at row j and column q, while p indicates the number of independent variables.
3. Consider the smallest distance k sets when sorting the distances between the missing rows and the complete rows. The imputation estimate is then calculated using their average.

2.3. Stochastic regression imputation

Stochastic regression imputation (SR) is a type of regression imputation. The regression equation derived from the observed data is used to determine the imputed missing value. To put it another way, a regression equation is constructed utilizing observed data y^* on x^* and then used to anticipate missing values on on y [12, 24]. The regression equation is as follows $\hat{y}^* = \hat{\beta}_0 + \hat{\beta}_1 x_1^* + ... + \hat{\beta}_p x_p^*$. The imputed values will, of course, fall on a regression line, showing an unrealistic association between the predictors and the outcome variable. To make the imputed value more realistic, a residual term is added. This is known as stochastic regression imputation (SR) [24].

The residual has a normal distribution with a mean of zero and a variance equal to the residual variance from the predictor's regression on the outcome [4, 12]. This is done to keep the data's variability.

2.4. Predictive mean matching imputation

Predictive mean matching imputation (PMM) is a semi-parametric imputation strategy that the mice R package recommends as the default imputation method to act as a baseline method for comparison. It is frequently used in survey sampling to account for non-response items [9, 11, 25, 26].

When calculating missing values for a particular variable, predictive mean matching imputation takes a sample from all full cases (donors) of the observed values for that variable and tries to match projected values as closely as feasible. As a result, since this approach only uses seen data, its imputed values are clearly reasonable [25].

The assumption for PMM is that the distribution of missing values is the same as that of the donors' observed data. Imputations outside the observed data range will not definitely occur and it can handle a variety of data types [24, 25].

The PMM process can be divided into the following 5 steps [26].

1. Estimate multiple linear regression coefficients $\hat{\beta}$ from complete cases.

2. Find a new set of regression coefficients β^* from the posterior predictive distribution of $\hat{\beta}$ obtained in step 1.

3. For predicted values for observable x or y, use $\hat{\beta}$ and for missing x or y, use β^*.

4. Find the closest (usually three) predicted values among the observed x_j or y_j, $j = m+1, ..., n$ for each missing x_i or y_i, $i = 1, ..., m$. Choose one of the three close cases at random and use the observed value of this instance to infer the missing value x_i or y_i.

5. Repeat steps 1–4 several times. However, the number of repetitions depends on the number of missing data and missing percentage.

2.5. Random forest imputation

Random forest imputation (RF) is one of the most commonly used feature selection methods because of its ease of use. The RF is a machine learning based on imputation approach for continuous or categorical data and can be used for classification and regression [9, 10]. R tool for missing data imputation based on RF "missForest" is defined as an imputation approach for mixed continuous and categorical data in the context of complicated interactions and nonlinearity that does not require the specification of variable distributions [9, 21, 22]. The RF can also accept categorical data, or continuous data as inputs without the need for dummy variables or scaled data [3].

Random forest imputation works by fitting several decision trees with random subsets of the data and then averaging over them to make a final prediction. As a result, random forest models have a high prediction accuracy [3]. The RF can be divided into 5 steps as follows Hong and Lynn [9]:

1. Replace the missing values with the mode (categorical variable) or the mean (continuous variable) of that variable.

2. Separate the dataset's observations into two categories: observable observations and missing observations. The training set consists of observed observations, while the prediction set consists of missing observations.

3. Create a random forest model from the data in 2).

4. The missing value in prediction set is estimated by the random forest model in 3).

5. Repeat steps 2–4 several times until $NRMSE_t$ is greater than $NRMSE_{t-1}$, $NRMSE$ is given by

$$NRMSE = \sqrt{\text{mean}((y_{true} - y_{imp})^2)/var(y_{true})},$$

where y_{true} denotes the missing y and has been replaced with mean or mode, y_{imp} denotes the estimate y from the random forest model and $var(y_{true})$ denotes the variance of y_{true}.

2.6. Stochastic regression random forest with equivalent weight

It is a method proposed by Thongsri and Samart [24] which is a combination of SR and RF imputations.

The SREW imputed value is given by $\hat{y}_{SREW} = W_M(\hat{y}_{SR} + \hat{y}_{RF})$, where the equivalent weight $W_M = \dfrac{1}{M}$ and M is the number of combination methods, $\hat{y}_{SR}$ is SR imputed value, $\hat{y}_{RF}$ is the RF imputed value and $\hat{y}_{SREW}$ is the estimated missing values from the SREW method.

2.7. *K-nearest random forest with equivalent weighted imputation*

Because the results of our trials shown that both KNN and RF perform well, we developed another composite approach, K-nearest random forest with equivalent weighted imputation (KREW), which is a combination of KNN and RF imputations.

The KREW imputed value is given by $\hat{y}_{KREW} = W_M(\hat{y}_{KNN} + \hat{y}_{RF})$, where the equivalent weight $W_M = \frac{1}{M}$, M is the number of combination methods, $\hat{y}_{KNN}$ is the KNN imputed, $\hat{y}_{RF}$ is the RF imputed value and $\hat{y}_{KREW}$ is the estimated missing values from the KREW method.

2.8. *K-nearest stochastic regression and random forest with equivalent weighted imputation*

To determine whether increasing the number of alternatives improves imputation performance, we propose a new three-component imputation method: k-nearest stochastic regression and random forest with equivalent weighted imputation (KSREW), which is a combination of KNN, SR, and RF methods with equivalent weight.

The KSREW imputed value is given by $\hat{y}_{KSREW} = W_M(\hat{y}_{KNN} + \hat{y}_{SR} + \hat{y}_{RF})$, where the equivalent weight $W_M = \frac{1}{M}$, M is the number of combination methods, $\hat{y}_{KNN}$ is the KNN imputed value, $\hat{y}_{SR}$ is the SR imputed value, $\hat{y}_{RF}$ is the RF imputed value. and $\hat{y}_{KSREW}$ is the estimated missing values from KSREW method.

3. SIMULATION STUDY

We present the simulation studies that were conducted to compare the performance of the eight missing data approaches. The eight techniques were evaluated using the average mean squared error (AMSE) of the estimations from multiple linear regressions. The simulation research used a random sample of $n = 30, 60, 100$ and 150 and two independent variables' values were produced individually from a uniform distribution with $X_1 \sim U(3, 5)$ and $X_2 \sim U(1, 10)$. The corresponding values of Y are then given by $y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i; \quad i = 1, 2, ..., n$, where the true value of the coefficients $\beta_0 = \beta_1 = \beta_2 = 1$ and the random error term ε_i were derived from a normal distribution with a mean of 0 and a standard deviation of 5.

The missing data on the dependent variable Y_i and independent variables X_1, X_2 was generated for each sample size using the MCAR, MAR, and MNAR mechanisms. The percentages of missing data were 10%, 20%, 30%, and 40%. The eight approaches were then used to deal with the missing data, and the estimates of regression coefficients were produced. The simulation procedure was repeated $N = 1,000$ times. After that, the average mean squared error (AMSE) of the eight techniques was obtained. R software was used to run all simulations.

Table 1 shows the AMSE of the eight techniques when the missing data type is MCAR. The results reveal that the KSREW performs best when the sample size is small ($n < 100$), but the KNN appears to perform better when the sample size is large ($n > 100$). When $n < 60$, the SREW fares well next to the KSREW, but when $n > 60$ the KREW appears to perform better than the SREW.

Tables 2 and 3 show the AMSE of the eight techniques where the missing data types are MAR and MNAR, respectively. According to the results, the KSREW is the most efficient at estimating missing data. When $n < 60$, the SREW performs well next to the KSREW, but when $n > 60$, the KREW appears to perform better than the SREW.

We also developed and compared a four-composite imputation approach, which was not presented here. It was discovered that increasing the number of components of a composite approach does not always improve it. Furthermore, when the standard deviation of error was increased to 10 and 15, the findings matched the simulation with a standard deviation of error of 5.

Table 1. AMSE of the eight methods when the missing type is MCAR

Sample size	Missing data (%)	Methods							
		HD	KNN	PMM	SR	RF	SREW	KREW	KSREW
30	10	23.0131	22.3739	22.5430	22.4789	22.3504	22.3335	22.3430	**22.3144**
	20	23.5297	22.2472	22.6694	22.4022	22.2147	22.1569	22.1919	**22.1243**
	30	24.4512	22.6352	23.0765	22.7625	22.4835	22.3812	22.4907	**22.3627**
	40	24.7862	22.3359	23.0109	22.5600	22.2012	22.0354	22.1837	**21.9995**
60	10	24.1830	23.6165	23.7590	23.7342	23.6020	23.6046	23.5948	**23.5858**
	20	24.9708	23.6450	24.0184	23.8957	23.6554	23.6327	23.6167	**23.5817**
	30	25.4828	23.6175	24.1530	23.9636	23.6045	23.5784	23.5647	**23.5159**
	40	24.7476	23.1684	23.8303	23.6562	23.1616	23.1236	23.0928	**23.0269**
100	10	24.6037	23.9953	24.1694	24.1361	24.0166	24.0148	23.9932	**23.9873**
	20	25.2147	24.0081	24.3850	24.3413	24.0464	24.0679	**23.9984**	24.0026
	30	25.6509	23.6529	24.2192	24.1293	23.7332	23.7301	23.6500	**23.6349**
	40	26.1524	23.4885	23.2667	24.1101	23.5397	23.5548	23.4512	**23.4325**
150	10	24.9560	**24.2762**	23.4599	24.4343	24.3059	24.3074	24.2796	24.2781
	20	25.4918	**24.1681**	24.5181	24.5454	24.2436	24.2699	24.1804	24.1958
	30	25.8645	**23.8426**	24.4751	24.4276	23.9664	24.0073	23.8652	23.8892
	40	26.5881	23.8567	23.6921	24.6046	23.9630	24.0224	**23.8432**	23.8660

Table 2. AMSE of the eight methods when the missing type is MAR

Sample size	Missing data (%)	Methods							
		HD	KNN	PMM	SR	RF	SREW	KREW	KSREW
30	10	23.6187	22.9717	23.1170	23.0190	22.9453	22.9005	22.9384	**22.8946**
	20	23.9252	22.6334	22.9780	22.6490	22.6012	22.4535	22.5727	**22.4463**
	30	24.7880	22.9536	23.3720	22.9500	22.7679	22.6000	22.7880	**22.6059**
	40	25.1351	23.0633	23.5320	22.9990	22.7185	**22.5075**	22.7828	22.5292
60	10	24.3465	23.7020	23.8330	23.8140	23.7084	23.6924	23.6897	**23.6709**
	20	25.1408	23.8230	24.0860	23.9370	23.7706	23.7095	23.7611	**23.6916**
	30	25.4314	23.5791	23.9760	23.7620	23.4907	23.4039	23.4797	**23.3756**
	40	25.9619	23.7241	24.2930	23.8440	23.5425	23.4121	23.5499	**23.3943**
100	10	24.7894	24.0988	24.2460	24.2080	24.1194	24.0967	24.0956	**24.0761**
	20	25.2006	23.8864	24.1790	24.0940	23.9060	23.8697	23.8672	**23.8304**
	30	25.6545	23.7735	24.1830	24.0360	23.7233	23.6772	23.6995	**23.6342**
	40	26.4899	23.9778	24.5020	24.2670	23.8572	**23.7796**	23.8374	23.7309
150	10	25.0274	24.3197	24.4600	24.4090	24.3451	24.3147	24.3217	**24.2983**
	20	25.0986	24.1885	24.4730	24.3940	24.2147	24.1730	24.1744	**24.1348**
	30	25.5646	24.1325	24.5650	24.4360	24.1307	24.0885	24.0878	**24.0338**
	40	26.5930	24.1401	24.6000	24.5000	24.0202	23.9988	24.0022	**23.9339**

4. APPLICATION TO REAL LIFE DATA

We applied all of the investigated imputation approaches to a real-world dataset, the Algerian forest fires data set. The data was obtained from the UCI Machine Learning Repository [5]. The relationship between temperature, relative humidity, and fine fuel moisture from the Fire Weather Index (FWI) system was determined using multiple regression analysis. The results are shown in Tables 4–6.

Table 3. AMSE of the eight methods when the missing type is MNAR

Sample size	Missing data (%)	Methods							
		HD	KNN	PMM	SR	RF	SREW	KREW	KSREW
30	10	22.9858	22.3736	22.5360	22.4250	22.3438	22.3110	22.3389	**22.3032**
	20	24.0973	22.8261	23.0840	22.9090	22.7701	22.6777	22.7596	**22.6634**
	30	24.2715	22.4941	22.8940	22.4960	22.3226	**22.1493**	22.3431	22.1619
	40	25.4276	22.9561	23.6630	23.0910	22.7091	22.5529	22.7535	**22.5398**
60	10	24.2869	23.6400	23.8050	23.7410	23.6487	23.6243	23.6298	**23.6046**
	20	25.1465	23.8162	24.1310	23.9620	23.8159	23.7483	23.7829	**23.7187**
	30	25.7999	23.8666	24.2790	24.0600	23.8022	23.7114	23.7854	**23.6080**
	40	26.0995	23.5229	24.1030	23.7920	23.3513	23.2821	23.3575	**23.2430**
100	10	24.8889	24.2271	24.3730	24.3360	24.2483	24.2282	24.2249	**24.2069**
	20	25.5078	24.1565	24.4570	24.3970	24.1738	24.1559	24.1365	**24.1102**
	30	25.8731	23.8861	24.3180	24.2120	23.8499	23.8317	23.8242	**23.7782**
	40	26.4051	23.7853	24.3490	24.2040	23.6918	23.6742	23.6686	**23.6041**
150	10	25.1924	24.4706	24.6470	24.6160	24.4996	24.4931	24.4740	**24.4658**
	20	25.7579	24.3796	24.7560	24.6900	24.4099	24.4255	**24.3686**	24.3689
	30	26.4918	24.3998	24.9300	24.7910	24.4373	24.4145	24.3765	**24.3413**
	40	26.7831	24.0533	24.6690	24.6220	24.0404	24.0636	23.9777	**23.9552**

Tables 4–6 reveal that when the sample size is small, the KSREW technique performs effectively. However, when the sample size is large, it appears that the KNN outperforms the KSREW. These findings are consistent with those obtained in the simulation study, especially when the missing mechanism is MCAR.

Table 4. AMSE of the eight methods when the missing type is MCAR in Algerian forest fires data set

Sample size	Missing data (%)	Methods							
		HD	KNN	PMM	SR	RF	SREW	KREW	KSREW
30	10	5.8638	5.5105	5.5816	5.5634	5.5070	5.5065	5.5005	**5.4964**
	20	6.2372	5.5471	5.6914	5.6422	5.5414	5.5381	5.5275	**5.5186**
	30	6.4434	5.4121	5.6409	5.5375	5.3810	5.3746	5.3700	**5.3504**
	40	6.9455	5.5589	5.8729	5.7364	5.2760	5.5225	5.5106	**5.4901**
60	10	6.1524	5.7846	5.8452	5.8472	5.7944	5.7970	5.7841	**5.7831**
	20	6.4858	5.7503	5.8563	5.8845	5.7627	5.7740	5.7479	**5.7433**
	30	6.7916	5.7148	5.9115	5.9109	5.7343	5.7456	5.7067	**5.7053**
	40	7.1424	5.6378	5.9151	5.9270	5.6807	5.7011	5.6403	**5.6307**
100	10	6.0704	**5.7370**	5.9572	6.0116	5.8140	5.8653	5.8008	5.7915
	20	6.3565	**5.7245**	5.9823	6.0385	5.8413	5.8903	5.9735	5.8380
	30	6.5690	**5.7776**	5.9927	6.0457	5.8674	5.9907	5.8478	5.8453
	40	6.8938	5.8337	6.0224	6.0849	5.9817	6.2480	5.9667	**5.8104**
150	10	6.3290	**5.8008**	6.0188	6.0449	5.9740	5.9860	5.9599	5.9555
	20	6.6461	**5.8104**	6.0138	6.0731	5.9317	5.9529	5.8977	5.8863
	30	6.9370	**5.7890**	5.9965	6.0702	5.8715	5.8969	5.8543	5.8135
	40	7.3090	**5.7602**	6.0246	6.1350	5.8634	5.8998	5.9735	5.7854

Table 5. AMSE of the eight methods when the missing type is MAR in Algerian forest fires data set

Sample size	Missing data (%)	Methods							
		HD	KNN	PMM	SR	RF	SREW	KREW	KSREW
30	10	5.9446	5.6083	5.6526	5.6350	5.5830	5.5812	5.5877	**5.5789**
	20	6.2926	5.6017	5.7126	5.6395	5.5404	5.5325	5.5547	**5.5309**
	30	6.5617	5.5679	5.7319	5.5891	5.4735	5.4428	5.4924	**5.4458**
	40	7.0674	5.6412	5.9349	5.6995	5.5576	5.5149	5.5662	**5.0575**
60	10	6.2532	5.8982	5.9409	5.9408	5.8870	5.8882	5.8867	**5.8829**
	20	6.4864	5.7892	5.8917	5.8723	5.7683	5.7700	5.7658	**5.7537**
	30	6.8640	5.7490	5.8821	5.8785	5.7316	5.7284	5.7213	**5.7067**
	40	7.1473	5.6997	5.8785	5.8473	5.6636	5.6555	5.6544	**5.6307**
100	10	6.0989	5.9434	5.9799	5.9970	5.9454	5.9479	5.9393	**5.9384**
	20	6.2861	5.8568	5.9526	5.9695	5.8629	5.8666	5.8487	**5.8464**
	30	6.9067	5.7957	5.9356	5.9679	5.8095	5.8162	5.7863	**5.7841**
	40	7.2971	5.7805	5.9682	6.0118	5.7974	5.8048	5.7650	**5.7605**
150	10	6.3297	**5.9608**	6.0106	6.0328	5.9787	5.9810	5.9690	5.9699
	20	6.6078	**5.8835**	5.9751	6.0134	5.9032	5.9069	5.8836	5.8851
	30	6.9619	**5.8202**	5.9696	6.0106	5.8645	5.8653	5.8273	5.8264
	40	7.2735	**5.7332**	5.9447	6.0123	5.7923	5.8032	5.7395	5.7445

Table 6. AMSE of the eight methods when the missing type is MNAR in Algerian forest fires data set

Sample size	Missing data (%)	Methods							
		HD	KNN	PMM	SR	RF	SREW	KREW	KSREW
30	10	5.8062	5.5393	5.5896	5.5772	5.5324	5.5318	5.5306	**5.5256**
	20	6.1339	5.5106	5.6092	5.5472	5.4712	5.4557	5.4780	**5.4527**
	30	6.5713	5.5461	5.7063	5.6066	5.4913	5.4683	5.4964	**5.4600**
	40	6.9920	5.5300	5.7837	5.6146	5.4763	5.4341	5.4714	**5.4200**
60	10	6.1430	5.8427	5.8850	5.8807	5.8426	5.8381	5.8383	**5.8316**
	20	6.4968	5.8133	5.9050	5.8940	5.8124	5.8020	5.8021	**5.7903**
	30	6.8323	5.7983	5.9586	5.9227	5.7975	5.7864	5.7833	**5.7644**
	40	7.1279	5.7112	5.9297	5.9045	5.7216	5.7160	5.6941	**5.6793**
100	10	6.7238	5.9522	6.0029	6.0062	5.9591	5.9592	5.9536	**5.9504**
	20	6.5707	5.8931	5.9788	5.9930	5.9028	5.9017	5.8869	**5.8838**
	30	6.9480	5.8874	6.0491	6.0572	5.9225	5.9193	5.8924	**5.8857**
	40	7.2750	5.8043	6.0150	6.0306	5.8478	5.8453	5.8072	**5.7996**
150	10	6.2856	5.9601	6.0101	6.0056	5.9711	5.9661	5.9628	**5.9578**
	20	6.5950	**5.9070**	6.0137	6.0262	5.9362	5.9361	5.9144	5.9123
	30	6.9910	**5.8875**	6.0521	6.0830	5.9352	5.9404	5.9010	5.9001
	40	7.3358	**5.8401**	6.0688	6.0985	5.9137	5.9147	5.8583	5.8577

5. CONCLUSION

The main aim of this study is to develop and compare the efficiency of eight techniques for dealing with missing data on dependent and independent variables in multiple linear regression. The estimates obtained by the composite imputation method: SREW, KREW, and KSREW outperform other approaches, according to our simulation results. The KSREW, in particular, excelled in practically every situation. Furthermore, when the sample size was less than 60, the SREW performed well next to the KSREW, but when the sample size was greater than 60, the KREW looked to perform better. The outcomes of the real life application were likewise aligned with those of the simulated study, especially when the missing mechanism is MCAR.

Overall, the proposed composite approaches outperformed the single ones, particularly a three-component method called KSREW. However, although not reported here, increasing the number of components to a four-component method did not improve the imputation performance.

Acknowledgments. This work was supported by the Faculty of Science Research Fund, Prince of Songkla University Contract no. 1-2564-02-006.

REFERENCES

1. H. Aguilera, C. Guardiola-Albert, and C. Serrano-Hidalgo, "Estimating extremely large amounts of missing precipitation data", J. of Hydroinformatics **22**, 578–592 (2020).
2. R. R. Andridge and R. J. Little, "A Review of hot deck imputation for survey non-response", International Statistical Review **78**, 40–64 (2010).
3. K. B. Best, J. M. Gilligan, H. Baroud, A. R. Carrico, K. M. Donato, B. A. Ackerly, and B. Mallick, "Random forest analysis of two household surveys can identify important predictors of migration in Bangladesh", J. of Computational Social Science **4**(1), 77–100 (2017).
4. P. Chaovanaphan and W. Chaimongkol, "A comparison of the estimation methods for missing data in sample survey", The Journal of Applied Science **16**(1), 60–73 (2017).
5. P. Cortez and A. D. Morais, A data mining approach to predict forest fires using meteorological data. `https://archive.ics.uci.edu/ml/datasets/Algerian+Forest+Fires+Dataset++`. Accessed 2022.
6. C. K. Enders, *Applied missing data analysis* (The Guilford Press, New York, 2010).
7. J. Han and M. Kamber, *Data mining: Concepts and techniques* (Morgan Kaufmann Publishers, San Francisco, 2012).
8. K. Hengpraprohm and S. Jungjit, "Missing Value Imputation Method using Ensemble Technique For Microarray Data", Information Technology J. **14**(2), 9–17 (2018).
9. S. Hong and H. S. Lynn, "Accuracy of random-forest-based imputation of missing data in the presence of non-normality, non-linearity, and interaction", BMC Medical Research Methodology **20**, 1–12 (2020).
10. H. Ozen and C. Bal, "A study on missing data problem in random Forest", OSMANGAZI J. of Medicine **42**(1), 103–109 (2020).
11. A. Jadhav, D. Pramod, and K. Ramanathan, "Comparison of performance of data imputation methods for numeric dataset", Applied Artificial Intelligence **33**, 913–933 (2019).
12. R. Lamjaisue, A. Thongteeraparp, and J. Sinsomboonthong, "Comparison of missing data estimation methods for the multiple regression analysis with missing at random dependent variable", Science and Technology J. **25**(5), 766–777 (2017).
13. R. J. Little, "Missing-data adjustments in large surveys", J. of Business & Economic Statistics **6**, 287–296 (1988).
14. R. J. Little and D. B. Rubin, *Statistical analysis with missing data* (John Wiley & Sons, Hoboken, 2002).
15. P. Muliwan, N. Chutiman, and P. Pue-on, "Development of Hot-deck Corrected Item Mean (HDD-CIM) for Estimating Missing Data", J. of Science and Technology Mahasarakham University **33**(2), 175–178 (2014).
16. T. A. Myers, "Goodbye listwise deletion: Presenting hot deck imputation as an easy and effective tool for handling missing data", Communication Methods and Measures **5**, 297–310 (2011).
17. M. Pauzi, N. Azifah, Y. B. Wah, S. M. Deni, N. A. Rahim, and S. Khatijah, "Comparison of Single and MICE Imputation Methods for Missing Values: A Simulation Study", Pertanika J. of Science & Technology **29**(2), 979–998 (2021).
18. D. B. Rubin, "Statistical matching using file concatenation with adjusted weights and multiple imputations", J. of Business & Economic Statistics **4**, 87–94 (1986).
19. K. Samart, N. Jansakul, and M. Chongcheawchamnan, "Exact bootstrap confidence intervals for regression coefficients in small samples", Communications in Statistics-Simulation and Computation **47**(10), 2953–2959 (2018).
20. J. L. Schafer and J. W. Graham, "Missing data: Our view of the state of the art", Psychological Methods **7**, 147–177 (2002).

21. D. J. Stekhoven and P. Buhlmann, "MissForest-non-parametric missing value imputation for mixed-type data", Bioinformatics **28**, 112–118 (2012).
22. F. Tang and H. Ishwaran,"Random forest missing data algorithms", Statistical Analysis and Data Mining **10**, 363–377 (2017).
23. R. Thinh, K. Samart, and N. Jansakul, "Linear regression models for heteroscedastic and non-normal data", SCIENCEASIA **46**(3), 353–360 (2020).
24. T. Thongsri and K. Samart, "Composite Imputation Method for the Multiple Linear Regression with Missing at Random Data", International J. of Mathematics and Computer Science **17**(1), 51–62 (2022).
25. S. van Buuren, *Flexible imputation of missing Data* (Chapman and Hall/CRC, Boca Raton, 2018).
26. G. Vink, L. E. Frank, J. Pannekoek, and S. van Buuren, "Predictive mean matching imputation of semicontinuous variables", Statistica Neerlandica **68**, 61–90 (2014).

Paper II

mlrpro: Perform Stepwise Regression with Verifying Assumptions and Identifying Possible Box-Cox Transformation

Thongsri T., and Samart K.

mlrpro: Perform Stepwise Regression with Verifying Assumptions and Identifying Possible Box-Cox Transformation

by Thidarat Thongsri, Klairung Samart

Abstract The *mlrpro* package is an intuitive regression analysis tool that is suitable for novice users. It is a built-in package that can fit the regression model, select independent variables, validate the assumptions of multiple linear regression, transform data using the Box-Cox transformation, and determine which regression model is the most suited. The regression coefficients, residuals, fitted values, and statistics related to regression, such as residual standard error, multiple R-squared, F-statistic, and so on, may all be obtained through the use of our *mlrpro* package. In addition to this, it provides visualization tools of the residuals plot, the normal Q-Q plot, and the lambda interval plot derived from Box-Cox transformations.

Introduction

The regression analysis is one of the most often used statistical tools, and it is used to examine the relationship that exists between a response and the independent variables (Kutner et al., 2005). In particular, it is a mathematical equation that illustrates the way in which a group of factors might explain an outcome and the way in which the outcome varies depending on each factor. It finds widespread application in a variety of domains, including commerce, education, medicine, and other fields. However, one drawback of regression analysis is that it is a very time-consuming and complicated process that is made up of a lot of distinct calculations and investigations.

In a multiple linear regression, the independent variables may be continuous, discrete, or categorical; however, the response variable must be continuous (Faraway, 2014). The number of independent variables in the regression model is given by the constant p. When $p = 1$, it is called simple linear regression. The model can be expressed as $Y_i = \beta_0 + \beta_1 X_{i1} + \varepsilon_i$. But when $p > 1$, it is called multiple linear regression (MLR).

The multiple linear regression model is given by

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + ... + \beta_p X_{ip} + \varepsilon_i \tag{1}$$

where Y is the dependent variable, X are the independent variables, β are the unknown regression coefficients, ε_i is a random error term which is normally distributed with mean $\mathrm{E}[\varepsilon_i] = 0$, constant variance $\mathrm{Var}(\varepsilon_i) = \sigma^2$ and the subscript i represents the ith observation.

Currently, a range of statistical applications is available for analyzing multiple linear regressions. R is a tool for statistical data analysis that may be downloaded for free. To investigate multiple regression in R, the *lm*() and *stepwise*() commands can be utilized. The outputs of the program must be examined to determine whether independent variables influence response variables at the specified level of significance and to determine whether multiple regression analysis assumptions are met. If the assumptions are not met, the data must be investigated and transformed possibly using the Box-Cox transformation. To effectively and reliably summarize and report the program's outputs, therefore, all users must have a solid grasp of statistics. Consequently, (Thongsri and Samart, 2022) have developed a *mlrpro* tool for evaluating multiple linear regression that simplifies and streamlines the entire procedure for users.

The *mlrpro* package (Thongsri and Samart, 2022) is a tool for multiple regression, select independent variables, check multiple linear regression assumptions and identify a possible transformation. This package gives the best multiple linear regression models, coefficients, fitted values, residuals and so on. Moreover, it computes the estimated lambda value and converts data by using Box-Cox transformation. In this study, we use the marketing data set, air quality data set and trees data set to illustrate how to obtain the regression coefficients and regression-related statistics. The marketing data set contains the impact of three advertising media (YouTube, Facebook and newspaper) on sales. Data are the advertising budget in thousands of dollars along with the sales (Kassambara, 2019).

The following is how this article is structured. In the section Statistical background, we describe the process for creating a multiple linear regression model, checking assumptions and Box-Cox transformation. Then, we illustrate the usage of the *mlrpro* package with the example data in the section Illustrations. Finally, our contributions are summarized in the section Summary and discussion.

Statistical background

Multiple linear regression model

Consider the equation (1), in matrix terms, the multiple linear regression model is given by:

$$Y = X\beta + \varepsilon. \tag{2}$$

The random vector Y has the expectation $\mathrm{E}[Y] = X\beta$ and the variance-covariance matrix $\mathrm{Var}(Y) = \sigma^2 I$. The least square method is frequently used to estimate the parameter β. This method necessitates taking into account the sum of the n squared deviations. This criterion is represented by the symbol Q.:

$$Q = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_{i1} - \beta_2 X_{i2} - ... - \beta_p X_{ip})^2 \tag{3}$$

The least squares estimators are the values of $\beta_0, \beta_1, ..., \beta_p$ that minimize Q. Let us denote the vector of the least squares estimated regression coefficients $b_0, b_1, ..., b_p$ as b. The least squares estimators for the multiple linear regression model are:

$$b = (X^\top X)^{-1}(X^\top Y) \tag{4}$$

where $b = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_p \end{bmatrix}$. The fitted values are represented by $\hat{Y} = Xb$ and the residual terms are denoted by $e = Y - Xb$. The vector of the fitted values $\hat{Y}$ can be expressed in terms of the hat matrix as follows:

$$\hat{Y} = HY = X(X^\top X)^{-1}X^\top Y \tag{5}$$

where $H = X(X^\top X)^{-1}X^\top$.

Sums of squares and mean squares

For analyzing variance in matrix terms, the sums of squares are:

$$SST = Y^\top Y - (\frac{1}{n})Y^\top JY \tag{6}$$

$$SSE = (Y - Xb)^\top (Y - Xb) \tag{7}$$

$$SSR = b^\top X^\top Y - (\frac{1}{n})Y^\top JY \tag{8}$$

where J is an nxn matrix of 1s, SST has $n-1$ degrees of freedom, SSE has $n-p-1$ degrees of freedom since $p+1$ parameters need to be estimated in the regression model. Finally, SSR has p degrees of freedom representing the number of X variables.

The results from the analysis of variance, as well as the mean squares MSR and MSE, are displayed in Table 1:

Table 1: ANOVA table for multiple linear regression model

Source of Variation	df	Sum of Square	Mean Square	F
Regression	p	SSR	$MSR = \frac{SSR}{p}$	$F = \frac{MSR}{MSE}$
Error	$n-p-1$	SSE	$MSE = \frac{SSE}{n-p-1}$	
Total	$n-1$	SST		

Test for regression relation

To determine if the response variable Y and the set of X variables have a regression relationship, the hypothesis testing is:

$H_0 : \beta_0 = \beta_1 = ... = \beta_p = 0$
H_1 : not all $\beta_k (k = 0, ..., p)$ equal zero.

The test statistic is $F_{stat} = \frac{MSR}{MSE}$.

If $F_{stat} \leq F(1-\alpha; p, n-p-1)$, conclude H_0 i.e. the response variable Y and the set of X variables do not have a regression relationship. If $F_{stat} > F(1-\alpha; p, n-p-1)$, conclude H_1 i.e. there is at least one independent variable X that has a regression relationship with the response variable Y.

Test for β_k

To determine whether X_k has a regression relationship with the response variable Y, the hypothesis testing is:

$H_0 : \beta_k = 0$
$H_1 : \beta_k \neq 0$.

The test statistic we may use is $t_{stat} = \frac{b_k}{s\{b_k\}}$ where $s\{b_k\}$ is the estimated standard error of b_k.

If $|t_{stat}| \leq t(1-\alpha/2; n-p-1)$, conclude H_0 i.e. there is a regression relationship between X_k and Y, otherwise conclude H_1.

In R, use the $lm()$ command to analyze and construct a regression model. $lm()$ is a function that is used to fit linear models, including multivariate ones. It is capable of performing regression, single stratum analysis of variance, and analysis of covariance (Chambers and Hastie, 1991; Wilkinson and Rogers, 1973).

Multiple linear regression: An example

```
> install.packages("datarium")
> library(datarium)
> data(marketing)
> Model.lm <- lm(sales ~ . , data = marketing)
> summary(Model.lm)

Call:
lm(formula = sales ~ ., data = marketing)

Residuals:
     Min       1Q   Median       3Q      Max
-10.5932  -1.0690   0.2902   1.4272   3.3951

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.526667   0.374290   9.422   <2e-16 ***
youtube      0.045765   0.001395  32.809   <2e-16 ***
facebook     0.188530   0.008611  21.893   <2e-16 ***
newspaper   -0.001037   0.005871  -0.177     0.86
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.023 on 196 degrees of freedom
Multiple R-squared:  0.8972,        Adjusted R-squared:  0.8956
F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

If the significance level is set at 0.05, the newspaper variable seems to have no influence on the response variable (sales). Therefore, a new regression model that takes the YouTube and Facebook variables into account is needed. As in the following example:

```
> Model.lm <- lm(sales ~ youtube + facebook, data = marketing)
> summary(Model.lm)

Call:
lm(formula = sales ~ youtube + facebook, data = marketing)

Residuals:
     Min       1Q   Median       3Q      Max
-10.5572  -1.0502   0.2906   1.4049   3.3994
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.50532    0.35339   9.919   <2e-16 ***
youtube      0.04575    0.00139  32.909   <2e-16 ***
facebook     0.18799    0.00804  23.382   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.018 on 197 degrees of freedom
Multiple R-squared:  0.8972,        Adjusted R-squared:  0.8962
F-statistic: 859.6 on 2 and 197 DF,  p-value: < 2.2e-16
```

In the preceding example, which is detailed in the following section, the *step*() command is used to simplify and facilitate the process of selecting the best regression model.

Stepwise regression

The number of feasible models for multiple linear regression is up to 2^p for p independent variables. Evaluating all of the various options can be a challenging task. To make the work easier, three strategies are commonly employed to select the best model: forward selection, backward deletion, and stepwise regression.

In forward selection, the variable with the highest correlation to the dependent variable is chosen first to be included in the model, and then the forward selection process proceeds. The operation is completed when there are no more statistically significant variables (Smith, 2018; Walczak, 2000).

Backward deletion starts with all of the variables in the equation and then eliminates one at a time that is not statistically significant. Continue until all of the model's independent variables are statistically significant (Smith, 2018; Walczak, 2000).

The stepwise regression method combines forward selection and backward deletion. First, by forward selection, enter the independent variable with the highest correlation to the response variable. Backward deletion is then used to determine whether the included variable should be excluded. This process is repeated until all of the model's independent variables are statistically significant (Smith, 2018; Walczak, 2000).

In R, to choose a model in a stepwise algorithm by AIC, use the command:

```
step(object, data, direction = c("both", "backward", "forward"))
```

where *object* represents a model of an appropriate class (mainly "lm" and "glm") and *direction* is the mode of stepwise search which can be one of "both", "backward", or "forward", with a default of "both" (Chambers and Hastie, 1991; Venables and Ripley, 2002).

Stepwise regression: An example

```
> Model.step <- step(lm(sales ~ . , data = marketing))
> summary(Model.step)

Call:
lm(formula = sales ~ youtube + facebook, data = marketing)

Residuals:
     Min       1Q   Median       3Q      Max
-10.5572  -1.0502   0.2906   1.4049   3.3994

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.50532    0.35339   9.919   <2e-16 ***
youtube      0.04575    0.00139  32.909   <2e-16 ***
facebook     0.18799    0.00804  23.382   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.018 on 197 degrees of freedom
```

```
Multiple R-squared:  0.8972,        Adjusted R-squared:  0.8962
F-statistic: 859.6 on 2 and 197 DF,  p-value: < 2.2e-16
```

This function returns independent factors that significantly affect the response variable. For this example, a regression model may be expressed as follows: $\hat{Y} = 3.5053 + 0.0458(\text{youtube}) + 0.1880(\text{facebook})$.

Checking assumptions

To apply the linear regression model effectively, four key assumptions must be verified: linearity, independence, homoscedasticity, and normality (Thinh et al., 2020).

This package verifies the homoscedasticity and normality assumptions of the multiple linear regression model. Creating a plot of standardized residuals vs predicted values or testing with Levene's Test from the *car*() library (car: Companion to Applied Regression) (Fox and Weisberg, 2018) is the fundamental method for determining whether homoscedasticity is satisfied. Levene's Test is a statistical study used to determine whether or not the variances of many groups are equal. The syntax of the Levene's Test function is $leveneTest(y, group, center = median, ...)$, where y is a response variable or residuals for the default method (lm or formula object), $group$ is a factor defining groups, and *center* is the name of a function to compute the center of each group; the default uses median to provide a more robust test (Fox, 2015; Fox and Weisberg, 2018). This test uses the null and alternative hypotheses listed below:

H_0 : The variance of the errors is constant (Homoscedastic).
H_1 : The variance of the errors is not constant (Heteroscedastic).

Additionally, the multiple linear regression assumes that the residuals of the model have a normal distribution. We can examine the assumption visually with Q-Q plots or formally with statistical tests such as the Shapiro-Wilk and Kolmogorov-Smirnov tests. The Shapiro-Wilk test is used to validate the assumption in this package. $shapiro.test(x)$ is the syntax for the Shapiro-Wilk test function, where x represents a numeric vector of data values (Royston, 1982a,b, 1995). The following null and alternative hypotheses are used in this test:

H_0 : The errors follow a normal distribution.
H_1 : The errors do not follow a normal distribution.

Checking assumptions: An example

```
> install.packages("car")
> library(car)
> error <- Model.step$residuals
> error.Group <- factor(error <= median(error))
> leveneTest(error, group = error.Group, center = median)

Levene's Test for Homogeneity of Variance (center = median)
       Df F value    Pr(>F)
group   1   28.13 3.019e-07 ***
      198
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> shapiro.test(error)

        Shapiro-Wilk normality test

data:  error
W = 0.91804, p-value = 4.19e-09
```

If the significance level is set to 0.05, the p-value of Levene's Test is 3.019×10^{-7} and the p-value of the Shapiro-Wilk normality test is 4.19×10^{-9} indicating that the variance of the errors is not constant (heteroscedastic) and that the errors do not have a normal distribution. When the assumption fails, data transformation using the Box-cox transformation method is required.

Box-Cox transformation

In regression applications, the Box-Cox transformation is a typical approach for transforming a non-normally distributed data set into one that is more normally distributed. The goal of this method is to determine a value for lambda such that the converted data is as close to normally distributed as possible without violating the normality or variance constancy of the error distributions (Box and Cox, 1964).

The original form of the Box-Cox transformation is given by:

$$y(\lambda) = \begin{cases} \frac{y^{\lambda}-1}{\lambda}, & \text{if } \lambda \neq 0; \\ \log y, & \text{if } \lambda = 0. \end{cases} \tag{9}$$

We can perform a Box-Cox transformation in R by using the Box-Cox function's syntax $boxcox()$ from the $MASS()$ library (Box and Cox, 1964). $boxcox(object, ...)$ offers the best lambda for the transformation, where *object* is a formula or fitted model object. The *mlrpro* package's data transformation is determined according to Table. 2.

Table 2: Common Box-Cox transformation

Lambda	Suitable
-2	$1/y^2$
-1	$1/y^1$
-0.5	$1/\sqrt{y}$
0	$\log y$
0.5	$\sqrt{y}$
1	y
2	y^2

Box-Cox transformation: An example

```
> install.packages("MASS")
> library(MASS)
> boxcox(sales ~ youtube + facebook, data = marketing)
```

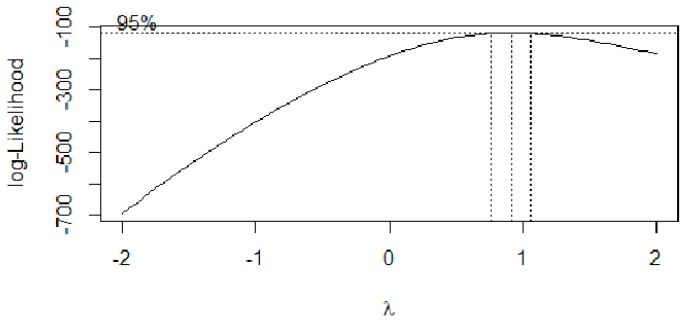

Figure 1: Box-Cox Transformation: log-likelihood as a function of λ values

The output of the Box-Cox transformation is shown in Figure 1. The dashed vertical line in the middle indicates the estimated parameter $\hat{\lambda}$ while the other two dashed lines represent the 95 percent confidence interval of lambda.

According to Figure 1, the optimal lambda value is around 0.9091, which can be extracted using the following R code:

```
> lambda.boxcox <- boxcox(sales ~ youtube + facebook, data = marketing)
> lambda.optimal <- subset(data.frame(lambda.boxcox),
  lambda.boxcox$y == max(lambda.boxcox$y))
> lambda.optimal$x
[1] 0.9090909
```

According to the estimated findings above and Figure 1, lambda has a value close to 1, indicating that no data transformation will take place. As a result, multiple linear regression may be inappropriate for this data.

Illustrations

The R package mlrpro

Our *mlrpro* package includes functions for fitting a regression model and selecting independent variables, converting data using the Box-Cox transformation, testing multiple linear regression assumptions, and identifying a possible model.

A set of standard methods is included with the *mlrpro* objects. As a result, inferences are simple, and the *mlrpro* package displays the estimates of regression coefficients, residual standard error, multiple R-squared, adjusted R-squared, F-statistic, and p-values.

The syntax of default function is, $mlrpro(Data, Y, Column\_Y, Alpha)$ and the four arguments of $mlrpro()$ function are described in Table 3.

Table 3: Description of $mlrpro()$ function arguments

Argument	Description
Data	a data frame containing the variables in the model
Y	the response variable
Column_Y	the response variable column
Alpha	significance level

The $mlrpro()$ class displays the regression model results and related statistics such as coefficients, residuals, fitted values, lambda, and so on. Table 4 presents a list of components of the class *mlrpro*.

Table 4: Components of the *mlrpro* class

Object	Description
coefficients	a named vector of coefficients
residuals	the residuals, that is response minus fitted values
fitted.values	the fitted mean values
rank	the numeric rank of the fitted linear model
df.residual	the residual degrees of freedom
call	the matched call
terms	the terms object used
model	if requested (the default), the model frame used
lambda	estimated lambda value utilized in the data transformation

The mlrpro package implementation in R

The use of *mlrpro* is demonstrated using examples, beginning with the marketing dataset.

```
> install.packages("mlrpro")
> library(mlrpro)
> data(marketing)
> Model1 <- mlrpro(Data = marketing,Y = marketing$sales, Column_Y = 4, Alpha = 0.05)

-------------------------
Stepwise regression model
-------------------------
```

```
Call:
lm(formula = y ~ youtube + facebook, data = Newdata, direct = "both")

Residuals:
     Min       1Q   Median       3Q      Max
-10.5572  -1.0502   0.2906   1.4049   3.3994

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.50532    0.35339   9.919   <2e-16 ***
youtube      0.04575    0.00139  32.909   <2e-16 ***
facebook     0.18799    0.00804  23.382   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.018 on 197 degrees of freedom
Multiple R-squared:  0.8972,        Adjusted R-squared:  0.8962
F-statistic: 859.6 on 2 and 197 DF,  p-value: < 2.2e-16

----------------------
Checking  Assumptions
----------------------
[1] The errors do not follow a normal distribution.
[1] The variance of the errors is not constant (Heteroscedastic).
------------------------
 Box cox transformation
------------------------
[1] Optimal lambda approximate to 1.
The multiple linear regression may not be appropriate for this data.
```

Given the above results, the estimated regression model is $\hat{Y} = 3.5053 + 0.0458(youtube) + 0.1880(facebook)$, with the multiple R-squared $(R^2) = 98.72\%$. The results show that the errors do not have a normal distribution and that the variance of the errors is not constant (heteroscedastic). Furthermore, based on the Box-Cox transformation, multiple linear regression may be inappropriate for this data.

We can extract the regression coefficients as follows:

```
> Model1$coefficients
(Intercept)     youtube    facebook
 3.50531989  0.04575482  0.18799423
```

The trees data set is another example of how to use *mlrpro()*. In this data set, the girth, height, and volume of timber in 31 felled black cherry trees are measured, with volume treated as a response variable and the rest as independent variables (Atkinson, 1987).

```
> data(trees)
> Model2 <- mlrpro(Data = trees,Y = trees$Volume, Column_Y = 3, Alpha = 0.01)
-------------------------
Stepwise regression model
-------------------------

Call:
lm(formula = y ~ ., data = Newdata1)

Residuals:
   Min     1Q Median     3Q    Max
-8.065 -3.107  0.152  3.495  9.587

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -36.9435     3.3651  -10.98 7.62e-12 ***
Girth         5.0659     0.2474   20.48  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 4.252 on 29 degrees of freedom
Multiple R-squared:  0.9353,        Adjusted R-squared:  0.9331
F-statistic: 419.4 on 1 and 29 DF,  p-value: < 2.2e-16

---------------------
Checking  Assumptions
--------------------
[1] The errors follow a normal distribution.
[1] The variance of the errors is constant (Homoscedastic).
```

The estimated regression model from the above results is $\hat{Y} = -36.9435 + 5.0659(Girth)$, with multiple R-squared $(R^2) = 93.53\%$. The results show that the errors have a normal distribution and that the variance of the errors is constant (homoscedastic). As a result, this model is suitable for this data set.

For the last example, we used the air quality data set. This data set consists of daily air quality measurements taken in New York from May to September 1973. A data frame with 153 observations on 6 variables, Ozone, Solar. R, Wind, Temp, Month, and Day were all recorded. Ozone is regarded as a response variable (Chambers et al., 1983).

```
> data(airquality)
> Model3 <- mlrpro(Data = airquality, Y = airquality$Ozone, Column_Y = 1, Alpha = 0.05)
-------------------------------------------------
Regression model derived from data transformations
-------------------------------------------------
The lambda value utilized in the data conversion is 0
and transformation of the dependent variable is in the form: y = log(y)

Call:
lm(formula = y ~ 0 + ., data = Newdata1)

Residuals:
    Min      1Q  Median      3Q     Max
-40.675 -15.446  -5.526  13.479  88.822

Coefficients:
        Estimate Std. Error t value Pr(>|t|)
Solar.R  0.06306    0.02387   2.641  0.00948 **
Wind    -4.59884    0.48653  -9.452 8.21e-16 ***
Temp     0.98525    0.08739  11.275  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 21.84 on 108 degrees of freedom
  (42 observations deleted due to missingness)
Multiple R-squared:  0.8383,        Adjusted R-squared:  0.8338
F-statistic: 186.7 on 3 and 108 DF,  p-value: < 2.2e-16

---------------------
Checking  Assumptions
--------------------
[1]  The errors do not follow a normal distribution.
[1]  The variance of the errors is not constant (Heteroscedastic).
```

The estimated regression model from the above results is $\hat{log\ y} = 0.0631(Solar.R) - 4.5988(Wind) + 0.9853(Temp)$. The results show that the errors do not follow a normal distribution and that the variance of the errors is not constant (heteroscedastic). This indicates that this model is not appropriate for the current data set. Further investigation by the data owner is required.

The residuals versus fitted values plot and normal Q-Q plot are also displayed in the *mlrpro* package as follows.

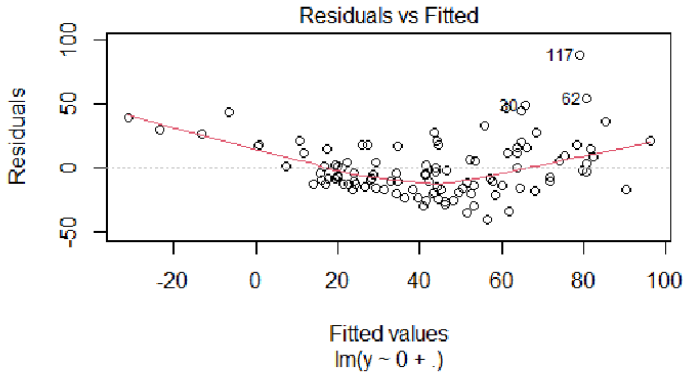

Figure 2: The residuals versus fitted values plot

If the residuals "bounce" around the residual = 0 line, this indicates that the relationship is likely to be linear. Furthermore, if the residuals scatter randomly with no pattern near 0, we can assume homoscedasticity. Figure 2 shows that the residuals are heteroscedastic due to the curve pattern.

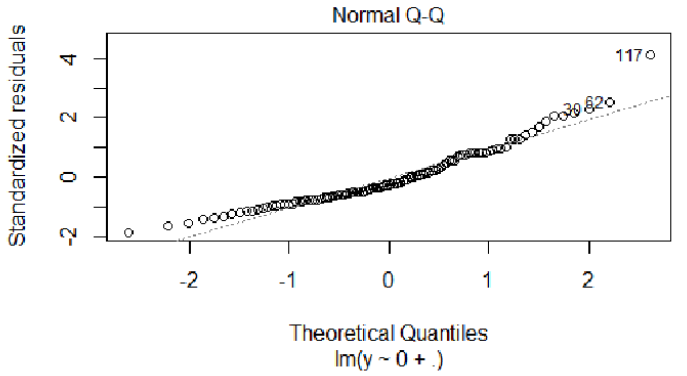

Figure 3: The normal Q-Q plot

The normal Q-Q plot is a tool used to determine whether errors have a normal distribution. The errors appear to follow a normal distribution in Figure 3 because all of the points are close to the straight line, but the Shapiro-Wilk test results show that the errors do not follow a normal distribution.

Summary and discussion

The method of regression analysis is complex and challenging, and it is necessary to have understanding of statistics in order to complete it. Because of this issue, we developed the *mlrpro* package, which is a statistical tool for the R software that can be downloaded for free and is designed to manage the entire process for those who are just starting out.

This package provides access to an extensive collection of open-source tools that can be used for carrying out regression analysis. The *mlrpro* package is a tool for conducting multiple regression, which can automatically select independent variables, validate multiple linear regression assumptions, and determine feasible models. Functions for calculating the best multiple linear regression models,

coefficients, fitted values, and residuals, among other things, have been developed by our team. In addition, the lambda value to determine the transformation may be computed using the Box-Cox transformation, and graphical displays of residuals against fitted values plot, normal Q-Q plot, and lambda interval plot can be presented.

Because it is simple and straightforward for users, the *mlrpro* package is ideal for users who are just starting out with regression analysis. Additionally, we expect that the accessibility of these tools will lead to an increase in their overall utilization.

Acknowledgments

This work was supported by the Faculty of Science Research Fund, Prince of Songkla University Contract no. 1-2564-02-006.

Bibliography

A. C. Atkinson. *Plots, transformations and regression*. Oxford: Oxford University Press, 1987. [p8]

G. Box and D. Cox. An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–252, 1964. [p6]

J. M. Chambers and T. J. Hastie. *Statistical models in S*. New York: Chapman & Hall/CPC, 1991. [p3, 4]

J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey. *Graphical methods for data analysis*. New York: Chapman & Hall/CPC, 1983. [p9]

J. J. Faraway. *Linear models with R*. New York: Chapman and Hall/CRC, 2014. [p1]

J. Fox. *Applied regression analysis and generalized linear models*. Thousand Oaks: Sage Publications, 2015. [p5]

J. Fox and S. Weisberg. *An R companion to applied regression*. Thousand Oaks: Sage publications, 2018. [p5]

A. Kassambara. *datarium: Data bank for statistical analysis and visualization*, 2019. URL https://CRAN.R-project.org/package=datarium. R package version 0.1.0. [p1]

M. H. Kutner, C. J. Nachtsheim, J. Neter, and W. Li. *Applied linear statistical models*. New York: McGraw-Hill/Irwin, 2005. [p1]

J. P. Royston. An extension of shapiro and wilk's w test for normality to large samples. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(2):115–124, 1982a. URL https://doi.org/10.2307/2347973. [p5]

J. P. Royston. Algorithm as 181: The w test for normality. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(2):176–180, 1982b. URL https://doi.org/10.2307/2347986. [p5]

J. P. Royston. Remark as r94: A remark on algorithm as 181: The w-test for normality. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 44(4):547–551, 1995. URL https://doi.org/10.2307/2986146. [p5]

G. Smith. Step away from stepwise. *Journal of Big Data*, 5:1–12, 2018. [p4]

R. Thinh, K. Samart, and N. Jansakul. Linear regression models for heteroscedastic and non-normal data. *ScienceAsia*, 46(3):353–360, 2020. URL 10.2306/scienceasia1513-1874.2020.047. [p5]

T. Thongsri and K. Samart. *mlrpro: Perform stepwise regression with verifying assumptions and identifying possible Box-Cox transformation*, 2022. URL https://CRAN.Rproject.org/package=mlrpro. R package version 0.1.2. [p1]

W. N. Venables and B. D. Ripley. *Modern applied statistics with S*. New York: Springer, 2002. [p4]

B. Walczak. *Wavelets in Chemistry*. Elsevier, 2000. [p4]

G. N. Wilkinson and C. E. Rogers. Symbolic descriptions of factorial models for analysis of variance. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 22(3):392–399, 1973. URL https://doi.org/10.2307/2346786. [p3]

Klairung Samart
Statistics and Applications Research Unit, Division of Computational Science,
Faculty ofScience, Prince of Songkla University
Thailand
klairung.s@psu.ac.th

SOURCE CODE

1. A set of instructions used to develop and compared the efficiency of eight imputation methods.

```
> Population <- 100000
> set.seed(112)
> x1 <- runif(Population,3,5); #x1
> x2 <- runif(Population,1,10); #x2
> e <- rnorm(Population,0,5)
> b0 <- 1; b1 <- 1; b2 <- 1
> y <- b0+(b1*x1)+(b2*x2)+e; #y
> MyData <- data.frame(y,x1,x2); #MyData
> SampleSize <- 30
> PercenMissing <- 0.4
> Repeat <- 1000
> Sum_MSE_KNN <- 0
> Sum_MSE_Stochas <- 0
> Sum_MSE_PMM  <- 0
> Sum_MSE_MissForest<- 0
> Sum_MSE_Stochas_MissForest <- 0
> Sum_MSE_KNN_MissForest <- 0
> Sum_MSE_KNN_Stochas_MissForest <- 0
> Sum_MSE_HotDeck <- 0
```

```
> Sum_MSE_KNN_Stochas <- 0
> i <- 1
> while (i <= Repeat) {
    > Sample <- MyData[sample(nrow(MyData),SampleSize),]; #Sample
    >  Missing <- ampute(Sample,prop = PercenMissing,mech = "MCAR")
    >  MissingData <- Missing$amp
    #---------------------------- Method  ----------------------------#
    > #install.packages("VIM")
    > #library(VIM)
    > Predict_Hot.Deck <- hotdeck(MissingData,variable=c("y","x1","x2"))
    > Fit_Hot.Deck <- lm(Predict_Hot.Deck[,1] ~
    Predict_Hot.Deck[,2]+Predict_Hot.Deck[,3])
    > MSE_Hot.Deck <- (sum((Fit_Hot.Deck$fitted.values-
    Sample[,1])^2))/SampleSize
    > Sum_MSE_HotDeck <- MSE_Hot.Deck+Sum_MSE_HotDeck
    #-------------------------- KNN  ---------------------------------#
    > Sqrt_M <- round(sqrt(SampleSize-(PercenMissing*SampleSize)))
    > Predict_KNN <- kNN(MissingData,variable = c("y","x1","x2"),k=Sqrt_M)
    > Fit_KNN <- lm(Predict_KNN[,1] ~ Predict_KNN[,2]+Predict_KNN[,3])
    > MSE_KNN<- (sum((Fit_KNN$fitted.values- Sample[,1])^2))/SampleSize
    > Sum_MSE_KNN <- MSE_KNN+Sum_MSE_KNN

    #-------------------------- Stochas  ----------------------------#
    > #install.packages("mice")
```

```
> #library(mice)

> Predict_Stochas <- mice(MissingData,method = "norm.nob")

> Data_Predict_Stochas <- complete(Predict_Stochas);#Data_Predict_Stochas

> Fit_Stochas <- lm(Data_Predict_Stochas[,1] ~Data_Predict_Stochas
[,2]+Data_Predict_Stochas[,3])

> MSE_Stochas <- (sum((Fit_Stochas$fitted.values-
Sample[,1])^2))/SampleSize

> Sum_MSE_Stochas <- MSE_Stochas+Sum_MSE_Stochas

#------------------------- pmm  ---------------------------#

> Predict_PMM   <- mice(MissingData,method = "pmm")

> Data_Predict_PMM  <- complete(Predict_PMM);#Data_Predict_PMM

> Fit_PMM <- lm(Data_Predict_PMM[,1] ~
Data_Predict_PMM[,2]+Data_Predict_PMM[,3])

> MSE_PMM <- (sum((Fit_PMM$fitted.values-Sample[,1])^2))/SampleSize

> Sum_MSE_PMM <- Sum_MSE_PMM+MSE_PMM

#------------------------- MissForest ----------------------------#

> #install.packages("missForest")

> #library(missForest)

> MissForest <- missForest(MissingData)

> Forest <-  MissForest$ximp

> Fit_MissForest <- lm(Forest[,1] ~  Forest[,2]+ Forest[,3])

> MSE_MissForest <- (sum((Fit_MissForest$fitted.values-
Sample[,1])^2))/SampleSize

> Sum_MSE_MissForest <-  Sum_MSE_MissForest + MSE_MissForest
```

```
#------------------------ KNN_MissForest -----------------------------#

> KNN_MissForest_Y <- (Predict_KNN[,1]+Forest[,1])/2

> KNN_MissForest_X1 <- (Predict_KNN[,2]+Forest[,2])/2

> KNN_MissForest_X2 <- (Predict_KNN[,3]+Forest[,3])/2

> Fit_KNN_MissForest <-
lm(KNN_MissForest_Y~KNN_MissForest_X1+KNN_MissForest_X2)

> MSE_KNN_MissForest <- (sum((Fit_KNN_MissForest$fitted.values-
Sample[,1])^2))/SampleSize

> Sum_MSE_KNN_MissForest <- Sum_MSE_KNN_MissForest +
MSE_KNN_MissForest

#------------------------ Stochas_MissForest -----------------------------#

> Stochas_MissForest_Y <- (Data_Predict_Stochas[,1]+Forest[,1])/2

> Stochas_MissForest_X1 <- (Data_Predict_Stochas[,2]+Forest[,2])/2

> Stochas_MissForest_X2 <- (Data_Predict_Stochas[,3]+Forest[,3])/2

> Fit_Stochas_MissForest <-
lm(Stochas_MissForest_Y~Stochas_MissForest_X1+Stochas_MissForest_X2)

> MSE_Stochas_MissForest  <- (sum((Fit_Stochas_MissForest$fitted.values-
Sample[,1])^2))/SampleSize

> Sum_MSE_Stochas_MissForest <-
Sum_MSE_Stochas_MissForest+MSE_Stochas_MissForest

#------------------------ KNN_MissForest_Stochas -----------------------------#

> KNN_MissForest_Stochas_Y <-
(Predict_KNN[,1]+Forest[,1]+Data_Predict_Stochas[,1])/3

> KNN_MissForest_Stochas_X1 <-
(Predict_KNN[,2]+Forest[,2]+Data_Predict_Stochas[,2])/3
```

```
        > KNN_MissForest_Stochas_X2 <-
        (Predict_KNN[,3]+Forest[,3]+Data_Predict_Stochas[,3])/3

        > Fit_KNN_Stochas_MissForest <- lm(KNN_MissForest_Stochas_Y
        ~KNN_MissForest_Stochas_X1+KNN_MissForest_Stochas_X2)

        > MSE_KNN_Stochas_MissForest  <-
        (sum((Fit_KNN_Stochas_MissForest$fitted.values-
        Sample[,1])^2))/SampleSize

        > Sum_MSE_KNN_Stochas_MissForest <-
        Sum_MSE_KNN_Stochas_MissForest+MSE_KNN_Stochas_MissForest

        #------------------------ KNN_Stochas ----------------------------#

        > KNN_Stochas_Y <- (Predict_KNN[,1]+Data_Predict_Stochas[,1])/2

        > KNN_Stochas_X1 <- (Predict_KNN[,2]+Data_Predict_Stochas[,2])/2

        > KNN_Stochas_X2 <- (Predict_KNN[,3]+Data_Predict_Stochas[,3])/2

        > Fit_KNN_Stochas <- lm(KNN_Stochas_Y
        ~KNN_Stochas_X1+KNN_Stochas_X2)

        > MSE_KNN_Stochas  <- (sum((Fit_KNN_Stochas$fitted.values-
        Sample[,1])^2))/SampleSize

        > Sum_MSE_KNN_Stochas <-  Sum_MSE_KNN_Stochas +
        MSE_KNN_Stochas

        i = i+1

}

> AVG_MSE_KNN <- Sum_MSE_KNN/Repeat;round(AVG_MSE_KNN,4)

> AVG_MSE_PMM  <- Sum_MSE_PMM/Repeat;round(AVG_MSE_PMM,4)

> AVG_MSE_Stochas <- Sum_MSE_Stochas/Repeat;round(AVG_MSE_Stochas,4)

> AVG_MSE_MissForest <-
  Sum_MSE_MissForest/Repeat;round(AVG_MSE_MissForest,4)
```

```
> AVG_MSE_KNN_MissForest <-
  Sum_MSE_KNN_MissForest/Repeat;round(AVG_MSE_KNN_MissForest,4)

> AVG_MSE_Stochas_MissForest <-
  Sum_MSE_Stochas_MissForest/Repeat;round(AVG_MSE_Stochas_MissForest,4)

> AVG_MSE_Sum_MSE_HotDeck <-
  Sum_MSE_HotDeck/Repeat;round(AVG_MSE_Sum_MSE_HotDeck,4)

> AVG_MSE_Sum_KNN_Stochas_MissForest <-
Sum_MSE_KNN_Stochas_MissForest/Repeat;round(AVG_MSE_Sum_KNN_Stocha
s_MissForest,4)

> AVG_MSE_Sum_KNN_Stochas <-
  Sum_MSE_KNN_Stochas/Repeat;round(AVG_MSE_Sum_KNN_Stochas,4)
```

2. A set of instructions to create a function for analyzing multiple linear regressions.

```
> #install.packages("dplyr");library(dplyr)
> #install.packages("car");library(car)
> #install.packages("MASS");library(MASS)
> mlrpro <- function(Data,Y,Column_Y,Alpha) {
> Newdata <- DataNewdata[Column_Y] <- NULL
> y <- Y
> Newdata <- data.frame(y,Newdata)
> fit <- suppressWarnings(step(lm(y~.,data=Newdata ,direct="both"),trace = 0))
> sumfit <- suppressWarnings(summary(fit))
> Number_Beta <- as.numeric(nrow(sumfit$coefficients))
> Decision <- 0
> i <- 1
> while (i <= Number_Beta) {
        > x <- ifelse(sumfit$coefficients[i,4]<=Alpha,"Sig","NoSig")
        > Decision[i] <- x
        > i = i+1
  }
> view <- data.frame(sumfit$coefficients)
> view <- mutate(view, Decision )
> NoSig <- subset(view,view$Decision == "NoSig")
> RowNoSig <- as.numeric(nrow(NoSig))
```

```
> Sig <-  subset(view,view$Decision == "Sig")

> RowSig <- as.numeric(nrow(Sig))

> if (RowSig <= 0) {

        > Newdata1 <- Newdata

        > Find.Lambda <- (boxcox(y~.,data= Newdata1))

        > Find.Lambda2 <- data.frame(Find.Lambda$x,Find.Lambda$y)

        > Find.Lambda3 <-
        subset(Find.Lambda2,Find.Lambda$y==max(Find.Lambda$y))

        > Optimal.lambda <- Find.Lambda3$Find.Lambda.x

        > Optimal.lambda.true <- Find.Lambda3$Find.Lambda.x

        > ifelse (Optimal.lambda >= 0.75 && Optimal.lambda < 1.5,
        Optimal.lambda <- 1,

        > Optimal.lambda <- Optimal.lambda)

> if (Optimal.lambda == 1 ) {

        > writeLines(c("-------------------------","Stepwise regression model ","----------
        ---------------"))

        > print(suppressWarnings(summary(fit)))

        > writeLines(c("----------------------","Checking  Assumptions","----------------
        ----"))

        > writeLines("[1] The errors do not follow a normal distribution.")

        > writeLines("[1] The variance of the errors is not constant
        (Heteroscedastic).")

        > writeLines("-----------------------\n Box cox transformation \n----------------
        ------")

        > writeLines("[1] Optimal lambda approximate to 1.")
```

```
        > message("The multiple linear regression may not be appropriate for this
        data.")
        > plot(fit,1)
        > plot(fit,2)
        > list(
                coefficients = fit$coefficients,
                residuals = fit$residuals,
                fitted.values = fit$fitted.values,
                rank = fit$rank,
                df.residual = fit$df.residual,
                call = fit$call,
                terms = fit$terms,
                model = fit$mode
        )
} else if (Optimal.lambda != 1) {
        > writeLines(c("-------------------------------------------------","Regression model
        derived from data transformations","-------------------------------------------------
        "))
> if (Optimal.lambda > -0.25 && Optimal.lambda < 0.25){
        > lambda <- 0
        > writeLines(c("The lambda value utilized in the data conversion is 0 ","and
        transformation the dependent variable is in the form: y=log(y)"))
        > y.prime <- log(y)
        > Newdata1$y <- NULL
```

```
> Newdata1 <- data.frame(y.prime,Newdata1)
> fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))
> sum_fit_end <- suppressWarnings(summary(fit_end))
} else if (Optimal.lambda >= 0.25 && Optimal.lambda < 0.75 ) {
> lambda <- 0.5
> writeLines(c("The lambda value utilized in the data conversion is 0.5",
"and transformation the dependent variable is in the form: y=sqrt(y)"))
> y.prime <- sqrt(y)
> Newdata1$y <- NULL
> Newdata1 <- data.frame(y.prime,Newdata1)
> fit_end <- suppressWarnings(lm(y.prime~., data=Newdata1))
> sum_fit_end <- suppressWarnings(summary(fit_end))
} else if (Optimal.lambda >= 1.5 && Optimal.lambda <= 2 ) {
> lambda <- 2
> writeLines(c("The lambda value utilized in the data conversion is 2 ","and
> transformation the dependent variable is in the form: y=y^2"))
> y.prime <- (y^2)
> Newdata1$y <- NULL
> Newdata1 <- data.frame(y.prime,Newdata1)
> fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))
> sum_fit_end <- suppressWarnings(summary(fit_end))
} else if (Optimal.lambda <= -0.25 && Optimal.lambda > -0.75 ) {
> lambda <- -0.5
```

```
        > writeLines(c("The lambda value utilized in the data conversion is -0.5 "," and transformation the dependent variable is in the form: y=1/sqrt(y)"))

        > y.prime <- 1/sqrt(y)

        > Newdata1$y <- NULL

        > Newdata1 <- data.frame(y.prime,Newdata1)

        > fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))

        > sum_fit_end <- suppressWarnings(summary(fit_end))

} else if (Optimal.lambda <= -0.75 && Optimal.lambda > -1.5 ) {

        > lambda <- -1

        > writeLines(c("The lambda value utilized in the data conversion is -1 ","and transformation the dependent variable is in the form: y=1/y"))

        > y.prime <- 1/y

        > Newdata1$y <- NULL

        > Newdata1 <- data.frame(y.prime,Newdata1)

        > fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))

        > sum_fit_end <- suppressWarnings(summary(fit_end))

} else if (Optimal.lambda <= -1.5 && Optimal.lambda >= -2 ) {

        > lambda <- -2

        > writeLines(c("The lambda value utilized in the data conversion is "," and transformation the dependent variable is in the form: y=1/y^2"))

        > y.prime <- 1/(y^2)

        > Newdata1$y <- NULL

        > Newdata1 <- data.frame(y.prime,Newdata1)

        > fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))
```

```
> sum_fit_end <- suppressWarnings(summary(fit_end))
}
> error <- fit_end$residuals
> error.Group <- factor(error<=median(error))
> Normal <- shapiro.test(error)
> variance <- leveneTest(error,group = error.Group)
> variance_p <- variance$`Pr(>F)`[1]
> plot(fit_end,1)
> plot(fit_end,2)
> print(suppressWarnings(summary(fit_end)))
> writeLines(c ("----------------------","Checking  Assumptions","----------
----"))
> ifelse(Normal$p.value <= Alpha,
print.noquote (" The errors do not follow a normal distribution."),
print.noquote (" The errors follow a normal distribution."))
> ifelse(variance_p <= Alpha,
print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
print.noquote(" The variance of the errors is constant (Homoscedastic)."))
> list(
        coefficients = fit_end$coefficients,
        residuals = fit_end$residuals,
        fitted.values = fit_end$fitted.values,
        rank = fit_end$rank,
        df.residual = fit_end$df.residual,
```

```
                call = fit_end$call,
                terms = fit_end$terms,
                model = fit_end$model,
                lambda = lambda
        ) } }
else {
        > beta0 <- sumfit$coefficients[1,4]
        > RowNoSig <- as.numeric(nrow(NoSig))
        > if (beta0 <= Alpha && RowNoSig > 0)  {
                > delete.Intercept <- row.names(Sig)
                > delete.Intercept <- delete.Intercept[-1]
                > Newdata1 <- Newdata[delete.Intercept]
        > while(RowNoSig > 0) {
                > Newdata1 <- Newdata[delete.Intercept]
                > Newdata1 <- data.frame(y,Newdata1)
                > fit1 <- suppressWarnings(lm(y~., data=Newdata1))
                > sumfit1 <- suppressWarnings(summary(fit1))
                > Number_Beta1 <- as.numeric(nrow(sumfit1$coefficients))
        Decision1 <- 0
        > i <- 1
        > while( i <= Number_Beta1 ) {
                > x <- ifelse(sumfit1$coefficients[i,4]<= Alpha,"Sig","NoSig")
                > Decision1[i] <- x
```

```
        > i = i+1
}
> view1 <- data.frame(sumfit1$coefficients)
> view1 <- mutate(view1, Decision1)
> Sig <- subset(view1,view1$Decision == "Sig")
> delete.Intercept <- row.names(Sig)
> delete.Intercept <- delete.Intercept[-1]
> NoSig <- subset(view1,view1$Decision == "NoSig")
> RowNoSig <- as.numeric(nrow(NoSig))
> RowNoSig = RowNoSig+0
}
> error <- fit1$residuals
> error.Group <- factor(error<=median(error))
> Normal <- shapiro.test(error)
> variance <- leveneTest(error,group = error.Group)
> variance_p <- variance$`Pr(>F)`[1]
> if (Normal$p.value <= Alpha || variance_p <=Alpha) {
> Find.Lambda <- (boxcox(y~.,data= Newdata1))
> Find.Lambda2 <- data.frame(Find.Lambda$x,Find.Lambda$y)
> Find.Lambda3 <-
subset(Find.Lambda2,Find.Lambda$y==max(Find.Lambda$y))
> Optimal.lambda <- Find.Lambda3$Find.Lambda.x
> Optimal.lambda.true <- Find.Lambda3$Find.Lambda.x
```

```
> ifelse (Optimal.lambda >= 0.75 && Optimal.lambda < 1.5,Optimal.lambda
<- 1,

> Optimal.lambda <- Optimal.lambda )

> if (Optimal.lambda == 1 ) {

> writeLines("-------------------------\n Stepwise regression model \n----------
--------------")

> print(suppressWarnings(summary(fit1)))

> writeLines ("---------------------\n Checking Assumptions \n-------------------
--")

> writeLines("[1] The errors do not follow a normal distribution.")

> writeLines("[1] The variance of the errors is not constant
(Heteroscedastic).")

> writeLines("-----------------------\n Box cox transformation \n-----------------
------")

> writeLines("Optimal lambda approximate to 1.")

> message("The multiple linear regression may not be appropriate for this
data.")

> plot(fit1,1)

> plot(fit1,2)

> list(

        coefficients = fit1$coefficients,

        residuals = fit1$residuals,

        fitted.values = fit1$fitted.values,

        rank = fit1$rank,

        df.residual = fit1$df.residual,
```

```
            call = fit1$call,
            terms = fit1$terms,
            model = fit1$model
        )
} else if (Optimal.lambda != 1) {
        > writeLines(c("-------------------------------------------------","Regression model
        derived from data transformations","-------------------------------------------------
        "))
> if (Optimal.lambda > -0.25 && Optimal.lambda < 0.25){
        > lambda <- 0
        > writeLines(c("The lambda value utilized in the data conversion is 0","and
        transformation the dependent variable is in the form: y = log(y)"))
        > y.prime <- log(y)
        > Newdata1$y <- NULL
        > Newdata1 <- data.frame(y.prime,Newdata1)
        > fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))
        > sum_fit_end <- suppressWarnings(summary(fit_end))
        }
> else if (Optimal.lambda >= 0.25 && Optimal.lambda < 0.75 ) {
        > lambda <- 0.5
        > writeLines(c("The lambda value utilized in the data conversion is 0.5 ","and
        transformation the dependent variable is in the form: y=sqrt(y)"))
        > y.prime <- sqrt(y)
        > Newdata1$y <- NULL
```

```
> Newdata1 <- data.frame(y.prime,Newdata1)

> fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))

> sum_fit_end <- suppressWarnings(summary(fit_end))

} else if (Optimal.lambda >= 1.5 && Optimal.lambda <= 2 ) {

> lambda <- 2

> writeLines(c("The lambda value utilized in the data conversion is 2 ","and
transformation the dependent variable is in the form: y=y^2"))

> y.prime <- (y^2)

> Newdata1$y <- NULL

> Newdata1 <- data.frame(y.prime,Newdata1)

> fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))

> sum_fit_end <- suppressWarnings(summary(fit_end))

} else if (Optimal.lambda <= -0.25 && Optimal.lambda > -0.75 ) {

> lambda <- -0.5

> writeLines(c("The lambda value utilized in the data conversion is -0.5 ","and
transformation the dependent variable is in the form: y=1/sqrt(y)"))

> y.prime <- 1/sqrt(y)

> Newdata1$y <- NULL

> Newdata1 <- data.frame(y.prime,Newdata1)

> fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))

> sum_fit_end <- suppressWarnings(summary(fit_end))

} else if (Optimal.lambda <= -0.75 && Optimal.lambda > -1.5 ) {

> lambda <- -1
```

```
> writeLines(c("The lambda value utilized in the data conversion is -1 ","and t
ransformation the dependent variable is in the form: y=1/y"))
> y.prime <- 1/y
> Newdata1$y <- NULL
> Newdata1 <- data.frame(y.prime,Newdata1)
> fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))
> sum_fit_end <- suppressWarnings(summary(fit_end))
} else if (Optimal.lambda <= -1.5 && Optimal.lambda>= -2 ) {
> lambda <- -2
> writeLines(c("The lambda value utilized in the data conversion is -2","and
transformation the dependent variable is in the form: y=1/y^2"))
> y.prime <- 1/(y^2)
> Newdata1$y <- NULL
> Newdata1 <- data.frame(y.prime,Newdata1)
> fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))
> sum_fit_end <- suppressWarnings(summary(fit_end))
}
> Number_Beta <- as.numeric(nrow(sum_fit_end$coefficients))
> Decision <- 0
> i <- 1
> while (i <= Number_Beta) {
    > x <- ifelse(sum_fit_end$coefficients[i,4]<=Alpha,"Sig","NoSig")
    > Decision[i] <- x
    > i = i+1 }
```

```
        > view <- data.frame(sum_fit_end$coefficients)
        > view <- mutate(view, Decision )
        > NoSig <- subset(view,view$Decision == "NoSig")
        > RowNoSig <- as.numeric(nrow(NoSig))
        > Sig <-  subset(view,view$Decision == "Sig")
        > beta0 <- sum_fit_end$coefficients[1,4]
> if (beta0 <= Alpha && RowNoSig > 0) {
        > delete.Intercept <- row.names(Sig)
        > delete.Intercept <- delete.Intercept[-1]
        > Newdata1 <- Newdata[delete.Intercept]
> while(RowNoSig > 0) {
        > Newdata1 <- Newdata[delete.Intercept]
        > Newdata1 <- data.frame(y,Newdata1)
        > fit1 <- suppressWarnings(lm(y~., data=Newdata1))
        > sumfit1 <- suppressWarnings(summary(fit1))
        > Number_Beta1 <- as.numeric(nrow(sumfit1$coefficients))
        > Decision1 <- 0
        > i <- 1
        > while( i <= Number_Beta1 ) {
                > x <- ifelse(sumfit1$coefficients[i,4]<= Alpha,"Sig","NoSig")
                > Decision1[i] <- x
                > i = i+1}
        > view1 <- data.frame(sumfit1$coefficients)
```

```
> view1 <- mutate(view1, Decision1)

> Sig <- subset(view1,view1$Decision == "Sig")

> delete.Intercept <- row.names(Sig)

> delete.Intercept <- delete.Intercept[-1]

> NoSig <- subset(view1,view1$Decision == "NoSig")

> RowNoSig <- as.numeric(nrow(NoSig))

> RowNoSig = RowNoSig+0

}

> error <- fit1$residuals

> error.Group <- factor(error<=median(error))

> Normal <- shapiro.test(error)

> variance <- leveneTest(error,group = error.Group)

> variance_p <- variance$`Pr(>F)`[1]

> plot(fit1,1)

> plot(fit1,2)

> print(suppressWarnings(summary(fit1)))

> writeLines(c ("----------------------","Checking  Assumptions","----------
----"))

> ifelse(Normal$p.value <= Alpha,
print.noquote (" The errors do not follow a normal distribution."),
print.noquote (" The errors follow a normal distribution."))

> ifelse( variance_p <= Alpha,
print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
print.noquote(" The variance of the errors is constant (Homoscedastic)."))
```

```
        > list(
                coefficients = fit1$coefficients,
                residuals = fit1$residuals,
                fitted.values = fit1$fitted.values,
                rank = fit1$rank,
                df.residual = fit1$df.residual,
                call = fit1$call,
                terms = fit1$terms,
                model = fit1$model,
                lambda = lambda
        )
} else if (beta0 <= Alpha && RowNoSig <= 0) {
        > error_end <- fit_end$residuals
        > error.Group <- factor(error_end<=median(error_end))
        > Normal <- shapiro.test(error_end)
        > variance <- leveneTest(error_end,group = error.Group)
        > variance_p <- variance$`Pr(>F)`[1]
        > plot(fit_end,1)
        > plot(fit_end,2)
        > print(suppressWarnings(summary(fit_end)))
        > writeLines(c ("----------------------","Checking  Assumptions",  "----------
        ------"))
        > ifelse(Normal$p.value <= Alpha,
```

```
        print.noquote (" The errors do not follow a normal distribution."),
        print.noquote (" The errors follow a normal distribution."))

        > ifelse(variance_p <= Alpha,
        print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
        print.noquote(" The variance of the errors is constant (Homoscedastic)."))

        > list(

                coefficients = fit_end$coefficients,

                residuals = fit_end$residuals,

                fitted.values = fit_end$fitted.values,

                rank = fit_end$rank,

                df.residual = fit_end$df.residual,

                call = fit_end$call,

                terms = fit_end$terms,

                model = fit_end$model,

                lambda = lambda

        )
} else if (beta0 > Alpha && RowNoSig > 0) {

        > delete.Intercept <- row.names(Sig)

        > Newdata1 <- Newdata[delete.Intercept]

        > while(RowNoSig > 0) {

        > Newdata1 <- Newdata[delete.Intercept]

        > Newdata1 <- data.frame(y,Newdata1)

        > fit1 <- suppressWarnings(lm(y~0+.,data=Newdata1))

        > sumfit1 <- suppressWarnings(summary(fit1))
```

```
> Number_Beta1 <- as.numeric(nrow(sumfit1$coefficients))
> Decision1 <- 0
> i <- 1
> while( i <= Number_Beta1 ) {
        x <- ifelse(sumfit1$coefficients[i,4]<= Alpha,"Sig","NoSig")
        Decision1[i] <- x
        i = i+1}
> view1 <- data.frame(sumfit1$coefficients)
> view1 <- mutate(view1, Decision1)
> Sig <- subset(view1,view1$Decision == "Sig")
> delete.Intercept <- row.names(Sig)
> NoSig <- subset(view1,view1$Decision == "NoSig")
> RowNoSig <- as.numeric(nrow(NoSig))
> RowNoSig = RowNoSig+0
}
> error <- fit1$residuals
> error.Group <- factor(error<=median(error))
> Normal <- shapiro.test(error)
> variance <- leveneTest(error,group = error.Group)
> variance_p <- variance$`Pr(>F)`[1]
> plot(fit1,1)
> plot(fit1,2)
> print(suppressWarnings(summary(fit1)))
```

```
> writeLines(c ("----------------------",  "Checking  Assumptions","----------------
------"))

> ifelse(Normal$p.value <= Alpha,
print.noquote (" The errors do not follow a normal distribution.")
print.noquote (" The errors follow a normal distribution."))

> ifelse( variance_p <= Alpha,
print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
print.noquote(" The variance of the errors is constant (Homoscedastic)."))

> list(

        coefficients = fit1$coefficients,

        residuals = fit1$residuals,

        fitted.values = fit1$fitted.values,

        rank = fit1$rank,

        df.residual = fit1$df.residual,

        call = fit1$call,

        terms = fit1$terms,

        model = fit1$model,

        lambda = lambda

) } else {

> delete.Intercept <- row.names(Sig)

> Newdata1 <- Newdata[delete.Intercept]

> Newdata1 <- data.frame(y,Newdata1)

> fit1 <- suppressWarnings(lm(y~0+.,data=Newdata1))

> sumfit1 <- suppressWarnings(summary(fit1))
```

```
> error <- sumfit1$residuals

> error.Group <- factor(error<=median(error))

> Normal <- shapiro.test(error)

>  <- leveneTest(error,group = error.Group)

> variance_p <- variance$`Pr(>F)`[1]

> plot(fit1,1)

> plot(fit1,2)

> print(suppressWarnings(summary(fit1)))

> writeLines(c ("----------------------","Checking  Assumptions",  "----------
------"))

> ifelse(Normal$p.value <= Alpha,
print.noquote (" The errors do not follow a normal distribution."),
print.noquote (" The errors follow a normal distribution."))

> ifelse( variance_p <= Alpha,
print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
print.noquote(" The variance of the errors is constant (Homoscedastic)."))

> list(
        coefficients = fit1$coefficients,
        residuals = fit1$residuals,
        fitted.values = fit1$fitted.values,
        rank = fit1$rank,
        df.residual = fit1$df.residual,
        call = fit1$call,
```

```
        terms = fit1$terms,
        model = fit1$model,
        lambda = lambda
)}}
}else if (Normal$p.value >= Alpha && variance_p >= Alpha) {
> writeLines(c("------------------------","Stepwise regression model ","----------
--------------"))
> print(suppressWarnings(summary(fit1)))
> plot(fit1,1)
> plot(fit1,2)
> writeLines(c("----------------------","Checking  Assumptions","----------------
----"))
> writeLines("[1] The errors follow a normal distribution.")
> writeLines("[1] The variance of the errors is constant (Homoscedastic).")
> list(
        coefficients = fit1$coefficients,
        residuals = fit1$residuals,
        fitted.values = fit1$fitted.values,
        rank = fit1$rank,
        df.residual = fit1$df.residual,
        call = fit1$call,
        terms = fit1$terms,
        model = fit1$model
)}
```

```
} else if (beta0 <= Alpha && RowNoSig <= 0) {
        > delete.Intercept <- row.names(Sig)
        > delete.Intercept <- delete.Intercept[-1]
        > Newdata1 <- Newdata[delete.Intercept]
        > error <- sumfit$residuals
        > error.Group <- factor(error<=median(error))
        > Normal <- shapiro.test(error)
        > variance <- leveneTest(error,group = error.Group)
        > variance_p <- variance$`Pr(>F)`[1]
> if (Normal$p.value <= Alpha || variance_p <= Alpha) {
        > Newdata1 <- data.frame(y,Newdata1)
        > Find.Lambda <- (boxcox(y~., data=Newdata1))
        > Find.Lambda2 <- data.frame(Find.Lambda$x,Find.Lambda$y)
        > Find.Lambda3 <-
        subset(Find.Lambda2,Find.Lambda$y==max(Find.Lambda$y))
        > Optimal.lambda <- Find.Lambda3$Find.Lambda.x
        > Optimal.lambda.true <- Find.Lambda3$Find.Lambda.x
        > ifelse (Optimal.lambda >= 0.75 && Optimal.lambda < 1.5,Optimal.lambda
        <- 1,
        > Optimal.lambda <- Optimal.lambda)
> if (Optimal.lambda == 1 ) {
        > writeLines(c("------------------------","Stepwise regression model ","----------
        ----------"))
        > print(suppressWarnings(summary(fit)))
```

```
> writeLines(c("----------------------", "Checking Assumptions", "----------
-------"))

> writeLines("[1] The errors do not follow a normal distribution.")

> writeLines("[1] The variance of the errors is not constant
(Heteroscedastic).")

> writeLines("------------------------\n Box cox transformation \n----------
------")

> writeLines("[1] Optimal lambda approximate to 1.")

> message("The multiple linear regression may not be appropriate for this
data.")

> plot(fit,1)

> plot(fit,2)

> list(

        coefficients = fit$coefficients,

        residuals = fit$residuals,

        fitted.values = fit$fitted.values,

        rank = fit$rank,

        df.residual = fit$df.residual,

        call = fit$call,

        terms = fit$terms,

        model = fit$model

)

} else if (Optimal.lambda != 1) {
```

```
> writeLines(c("-------------------------------------------------","Regression model
derived from data transformations","-------------------------------------------------
"))
> if (Optimal.lambda > -0.25 && Optimal.lambda < 0.25){
lambda <- 0
> writeLines(c("The lambda value utilized in the data conversion is 0 ","and
transformation the dependent variable is in the form: y=log(y)"))
> y.prime <- log(y)
> Newdata1$y <- NULL
> Newdata1 <- data.frame(y.prime,Newdata1)
> fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))
> sum_fit_end <- suppressWarnings(summary(fit_end))
} else if (Optimal.lambda >= 0.25 && Optimal.lambda < 0.75 ) {
> lambda <- 0.5
> writeLines(c("The lambda value utilized in the data conversion is 0.5","and
transformation the dependent variable is in the form: y=sqrt(y)"))
> y.prime <- sqrt(y)
> Newdata1$y <- NULL
> Newdata1 <- data.frame(y.prime,Newdata1)
> fit_end <- suppressWarnings(lm(y.prime~., data=Newdata1))
> sum_fit_end <- suppressWarnings(summary(fit_end))
} else if (Optimal.lambda >= 1.5 && Optimal.lambda <= 2 ) {
>  lambda <- 2
```

```
        > writeLines(c("The lambda value utilized in the data conversion is 2 ","and
        transformation the dependent variable is in the form: y=y^2"))
        > y.prime <- (y^2)
        > Newdata1$y <- NULL
        > Newdata1 <- data.frame(y.prime,Newdata1)
        > fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))
        > sum_fit_end <- suppressWarnings(summary(fit_end))
} else if (Optimal.lambda <= -0.25 && Optimal.lambda > -0.75 ) {
        > lambda <- -0.5
        > writeLines(c("The lambda value utilized in the data conversion is -0.5 ","
        and transformation the dependent variable is in the form: y=1/sqrt(y)"))
        > y.prime <- 1/sqrt(y)
        > Newdata1$y <- NULL
        > Newdata1 <- data.frame(y.prime,Newdata1)
        > fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))
        > sum_fit_end <- suppressWarnings(summary(fit_end))
} else if (Optimal.lambda <= -0.75 && Optimal.lambda > -1.5 ) {
        > lambda <- -1
        > writeLines(c("The lambda value utilized in the data conversion is -1 ","and
        transformation the dependent variable is in the form: y=1/y"))
        > y.prime <- 1/y
        > Newdata1$y <- NULL
        > Newdata1 <- data.frame(y.prime,Newdata1)
        > fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))
```

```
> sum_fit_end <- suppressWarnings(summary(fit_end))
} else if (Optimal.lambda <= -1.5 && Optimal.lambda >= -2 ) {
> lambda <- -2
> writeLines(c("The lambda value utilized in the data conversion is "," and
transformation the dependent variable is in the form: y=1/y^2"))
> y.prime <- 1/(y^2)
> Newdata1$y <- NULL
> Newdata1 <- data.frame(y.prime,Newdata1)
> fit_end <- suppressWarnings(lm(y.prime~.,data=Newdata1))
> sum_fit_end <- suppressWarnings(summary(fit_end))
}
> Number_Beta <- as.numeric(nrow(sum_fit_end$coefficients))
> Decision <- 0
> i <- 1
> while (i <= Number_Beta) {
    > x <- ifelse(sum_fit_end$coefficients[i,4]<=Alpha,"Sig","NoSig")
    > Decision[i] <- x
    > i = i+1
    }
> view <- data.frame(sum_fit_end$coefficients)
> view <- mutate(view, Decision )
> NoSig <- subset(view,view$Decision == "NoSig")
> RowNoSig <- as.numeric(nrow(NoSig))
```

```
> Sig <-  subset(view,view$Decision == "Sig")
> beta0 <- sum_fit_end$coefficients[1,4]
> if (beta0 <= Alpha && RowNoSig > 0) {
> delete.Intercept <- row.names(Sig)
> delete.Intercept <- delete.Intercept[-1]
> Newdata1 <- Newdata[delete.Intercept]
> while(RowNoSig > 0) {
        > Newdata1 <- Newdata[delete.Intercept]
        > Newdata1 <- data.frame(y,Newdata1)
        > fit1 <- suppressWarnings(lm(y~., data=Newdata1))
        > sumfit1 <- suppressWarnings(summary(fit1))
        > Number_Beta1 <- as.numeric(nrow(sumfit1$coefficients))
        > Decision1 <- 0
        > i <- 1
> while( i <= Number_Beta1 ) {
        > x <- ifelse(sumfit1$coefficients[i,4]<= Alpha,"Sig","NoSig")
        > Decision1[i] <- x
        > i = i+1 }
> view1 <- data.frame(sumfit1$coefficients)
> view1 <- mutate(view1, Decision1)
> Sig <- subset(view1,view1$Decision == "Sig")
> delete.Intercept <- row.names(Sig)
> delete.Intercept <- delete.Intercept[-1]
```

```
> NoSig <- subset(view1,view1$Decision == "NoSig")

> RowNoSig <- as.numeric(nrow(NoSig))

> RowNoSig = RowNoSig+0

}

> error <- fit1$residuals

> error.Group <- factor(error<=median(error))

> Normal <- shapiro.test(error)

> variance <- leveneTest(error,group = error.Group)

> variance_p <- variance$`Pr(>F)`[1]

> plot(fit1,1)

> plot(fit1,2)

> print(suppressWarnings(summary(fit1)))

> writeLines(c ("----------------------","Checking  Assumptions","----------
----"))

> ifelse(Normal$p.value <= Alpha,
print.noquote (" The errors do not follow a normal distribution."),
print.noquote (" The errors follow a normal distribution."))

> ifelse( variance_p <= Alpha,
print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
print.noquote(" The variance of the errors is constant (Homoscedastic)."))

> list(

        coefficients = fit1$coefficients,

        residuals = fit1$residuals,

        fitted.values = fit1$fitted.values,
```

```
            rank = fit1$rank,
            df.residual = fit1$df.residual,
            call = fit1$call,
            terms = fit1$terms,
            model = fit1$model,
            lambda = lambda
        )
} else if (beta0 <= Alpha && RowNoSig <= 0) {
        > error_end <- fit_end$residuals
        > error.Group <- factor(error_end<=median(error_end))
        > Normal <- shapiro.test(error_end)
        > variance <- leveneTest(error_end,group = error.Group)
        > variance_p <- variance$`Pr(>F)`[1]
        > plot(fit_end,1)
        > plot(fit_end,2)
        > print(suppressWarnings(summary(fit_end)))
        > writeLines(c ("----------------------","Checking  Assumptions","----------
        ----"))
        > ifelse(Normal$p.value <= Alpha,
        print.noquote (" The errors do not follow a normal distribution."),
        print.noquote (" The errors follow a normal distribution."))
        > ifelse( variance_p <= Alpha,
        print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
        print.noquote(" The variance of the errors is constant (Homoscedastic)."))
```

```
        > list(
                coefficients = fit_end$coefficients,
                residuals = fit_end$residuals,
                fitted.values = fit_end$fitted.values,
                rank = fit_end$rank,
                df.residual = fit_end$df.residual,
                call = fit_end$call,
                terms = fit_end$terms,
                model = fit_end$model,
                lambda = lambda
        )
} else if (beta0 > Alpha && RowNoSig > 0) {
        > delete.Intercept <- row.names(Sig)
        > Newdata1 <- Newdata[delete.Intercept]
        > while(RowNoSig > 0) {
        > Newdata1 <- Newdata[delete.Intercept]
        > Newdata1 <- data.frame(y,Newdata1)
        > fit1 <- suppressWarnings(lm(y~0+.,data=Newdata1))
        > sumfit1 <- suppressWarnings(summary(fit1))
        > Number_Beta1 <- as.numeric(nrow(sumfit1$coefficients))
        > Decision1 <- 0
        > i <- 1
        > while( i <= Number_Beta1 ) {
```

```
        > x <- ifelse(sumfit1$coefficients[i,4]<= Alpha,"Sig","NoSig")
        > Decision1[i] <- x
        > i = i+1 }
> view1 <- data.frame(sumfit1$coefficients)
> view1 <- mutate(view1, Decision1)
> Sig <- subset(view1,view1$Decision == "Sig")
> delete.Intercept <- row.names(Sig)
> NoSig <- subset(view1,view1$Decision == "NoSig")
> RowNoSig <- as.numeric(nrow(NoSig))
> RowNoSig = RowNoSig+0
}
> error <- fit1$residuals
> error.Group <- factor(error<=median(error))
> Normal <- shapiro.test(error)
> variance <- leveneTest(error,group = error.Group)
> variance_p <- variance$`Pr(>F)`[1]
> plot(fit1,1)
> plot(fit1,2)
> print(suppressWarnings(summary(fit1)))
> writeLines(c ("----------------------","Checking  Assumptions","----------
----"))
> ifelse(Normal$p.value <= Alpha,
print.noquote (" The errors do not follow a normal distribution."),
print.noquote (" The errors follow a normal distribution."))
```

```
> ifelse( variance_p <= Alpha,
print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
print.noquote(" The variance of the errors is constant (Homoscedastic)."))
> list(
        coefficients = fit1$coefficients,
        residuals = fit1$residuals,
        fitted.values = fit1$fitted.values,
        rank = fit1$rank,
        df.residual = fit1$df.residual,
        call = fit1$call,
        terms = fit1$terms,
        model = fit1$model,
        lambda = lambda
)
} else {
> delete.Intercept <- row.names(Sig)
> Newdata1 <- Newdata[delete.Intercept]
> Newdata1 <- data.frame(y,Newdata1)
> fit1 <- suppressWarnings(lm(y~0+.,data=Newdata1))
> sumfit1 <- suppressWarnings(summary(fit1))
> error <- sumfit1$residuals
> error.Group <- factor(error<=median(error))
> Normal <- shapiro.test(error)
> variance <- leveneTest(error,group = error.Group)
```

```
> variance_p <- variance$`Pr(>F)`[1]

> plot(fit1,1)

> plot(fit1,2)

> print(suppressWarnings(summary(fit1)))

> writeLines(c ("----------------------","Checking  Assumptions","----------
----"))

> ifelse(Normal$p.value <= Alpha,
print.noquote (" The errors do not follow a normal distribution."),
print.noquote (" The errors follow a normal distribution."))

> ifelse( variance_p <= Alpha,
print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
print.noquote(" The variance of the errors is constant (Homoscedastic)."))

> list(

        coefficients = fit1$coefficients,

        residuals = fit1$residuals,

        fitted.values = fit1$fitted.values,

        rank = fit1$rank,

        df.residual = fit1$df.residual,

        call = fit1$call,

        terms = fit1$terms,

        model = fit1$model,

        lambda = lambda

) } }

} else if (Normal$p.value >= Alpha && variance_p >= Alpha) {
```

```
> plot(fit,1)

> plot(fit,2)

> writeLines(c("------------------------","Stepwise regression model ","----------
---------------"))

> print(suppressWarnings(summary(fit)))

> writeLines(c ("----------------------","Checking  Assumptions","----------------
----"))

> writeLines("[1] The errors follow a normal distribution.")

> writeLines("[1] The variance of the errors is constant (Homoscedastic).")

> list(

        coefficients = fit$coefficients,

        residuals = fit$residuals,

        fitted.values = fit$fitted.values,

        rank = fit$rank,

        df.residual = fit$df.residual,

        call = fit$call,

        terms = fit$terms,

        model = fit$model

) }

} else if (beta0 > Alpha && RowNoSig > 0) {

> delete.Intercept <- row.names(Sig)

> Newdata1 <- Newdata[delete.Intercept]

> while(RowNoSig > 0) {

> Newdata1 <- Newdata[delete.Intercept]
```

```
> Newdata1 <- data.frame(y,Newdata1)
> fit1 <- suppressWarnings(lm(y~0+.,data=Newdata1))
> sumfit1 <- suppressWarnings(summary(fit1))
> Number_Beta1 <- as.numeric(nrow(sumfit1$coefficients))
> Decision1 <- 0
> i <- 1
> while( i <= Number_Beta1 ) {
        > x <- ifelse(sumfit1$coefficients[i,4]<= Alpha,"Sig","NoSig")
        > Decision1[i] <- x
        > i = i+1 }
> view1 <- data.frame(sumfit1$coefficients)
> view1 <- mutate(view1, Decision1)
> Sig <- subset(view1,view1$Decision == "Sig")
> delete.Intercept <- row.names(Sig)
> NoSig <- subset(view1,view1$Decision == "NoSig")
> RowNoSig <- as.numeric(nrow(NoSig))
> RowNoSig = RowNoSig+0
}
> error <- fit1$residuals
> error.Group <- factor(error<=median(error))
> Normal <- shapiro.test(error)
> variance <- leveneTest(error,group = error.Group)
> variance_p <- variance$`Pr(>F)`[1]
```

```
> if (Normal$p.value <= Alpha || variance_p <= Alpha) {
    > Find.Lambda <- (boxcox(y~.,data=Newdata1))
    > Find.Lambda2 <- data.frame(Find.Lambda$x,Find.Lambda$y)
    > Find.Lambda3 <-
    subset(Find.Lambda2,Find.Lambda$y==max(Find.Lambda$y))
    > Optimal.lambda <- Find.Lambda3$Find.Lambda.x
    > Optimal.lambda.true  <- Find.Lambda3$Find.Lambda.x
    > ifelse (Optimal.lambda >= 0.75 && Optimal.lambda < 1.5,Optimal.lambda
    <- 1,
    > Optimal.lambda <- Optimal.lambda)
> if (Optimal.lambda == 1 ) {
    > writeLines(c("------------------------","Stepwise regression model ","----------
    ---------------"))
    > print(suppressWarnings(summary(fit1)))
    > writeLines(c("----------------------","Checking  Assumptions","----------------
    ----"))
    > writeLines("[1] The errors do not follow a normal distribution.")
    > writeLines("[1] The variance of the errors is not constant
    (Heteroscedastic).")
    > writeLines("-----------------------\n Box cox transformation \n----------------
    ------")
    > writeLines("Optimal lambda approximate to 1.")
    > message("The multiple linear regression may not be appropriate for this
    data.")
    > plot(fit1,1)
```

```
        > plot(fit1,2)
        > list(
                coefficients = fit1$coefficients,
                residuals = fit1$residuals,
                fitted.values = fit1$fitted.values,
                rank = fit1$rank,
                df.residual = fit1$df.residual,
                call = fit1$call,
                terms = fit1$terms,
                model = fit1$model
        )
} else if (Optimal.lambda!= 1) {
        > writeLines(c ("-------------------------------------------------","Regression
        model derived from data transformations","-----------------------------------------
        ---------"))
> if (Optimal.lambda> -0.25 && Optimal.lambda< 0.25){
        > lambda <- 0
        > writeLines(c("The lambda value utilized in the data conversion is 0","and
                transformation the dependent variable is in the form: y = log(y)"))
        > y.prime <- log(y)
        > Newdata1$y <- NULL
        > Newdata1 <- data.frame(y.prime,Newdata1)
        > fit_end <- suppressWarnings(lm(y.prime~0+.,data=Newdata1))
        > sum_fit_end <- suppressWarnings(summary(fit_end))
```

```
} else if (Optimal.lambda >= 0.25 && Optimal.lambda < 0.75 ) {
    > lambda <- 0.5
    > writeLines(c("The lambda value utilized in the data conversion is 0.5 ","and
    transformation the dependent variable is in the form: y=sqrt(y)"))
    > y.prime <- sqrt(y)
    > Newdata1$y <- NULL
    > Newdata1 <- data.frame(y.prime,Newdata1)
    > fit_end <- suppressWarnings(lm(y.prime~0+.,data= Newdata1))
    > sum_fit_end <- suppressWarnings(summary(fit_end))
} else if (Optimal.lambda >= 1.5 && Optimal.lambda <= 2 ) {
    > lambda <- 2
    > writeLines(c("The lambda value utilized in the data conversion is 2",   "and
    transformation the dependent variable is in the form: y=y^2"))
    > y.prime <- (y^2)
    > Newdata1$y <- NULL
    > Newdata1 <- data.frame(y.prime,Newdata1)
    > fit_end <- suppressWarnings(lm(y.prime~0+.,data=Newdata1))
    > sum_fit_end <- suppressWarnings(summary(fit_end))
} else if (Optimal.lambda <= -0.25 && Optimal.lambda > -0.75 ) {
    > lambda <- -0.5
    > writeLines(c("The lambda value utilized in the data conversion is -0.5","and
    transformation the dependent variable is in the form: y=1/sqrt(y)"))
    > y.prime <- 1/sqrt(y)
    > Newdata1$y <- NULL
```

```
        > Newdata1 <- data.frame(y.prime,Newdata1)
        > fit_end <- suppressWarnings(lm(y.prime~0+.,data=Newdata1))
        > sum_fit_end <- suppressWarnings(summary(fit_end))
} else if (Optimal.lambda <= -0.75 && Optimal.lambda > -1.5 ) {
        > lambda <- -1
        > writeLines(c("The lambda value utilized in the data conversion is -1","and
        transformation the dependent variable is in the form: y=1/y"))
        > y.prime <- 1/y
        > Newdata1$y <- NULL
        > Newdata1 <- data.frame(y.prime,Newdata1)
        > fit_end <- suppressWarnings(lm(y.prime~0+.,data =  Newdata1))
        > sum_fit_end <- suppressWarnings(summary(fit_end))
} else if (Optimal.lambda <= -1.5 && Optimal.lambda >= -2 ) {
        > lambda <- -2
        > writeLines(c("\n","The lambda value utilized in the data conversion is -2 ","
        and transformation the dependent variable is in the form: y=1/y^2"))
        > y.prime <- 1/(y^2)
        > Newdata1$y <- NULL
        > Newdata1 <- data.frame(y.prime,Newdata1)
        > fit_end <- suppressWarnings(lm(y.prime~0+.,data =  Newdata1))
        > sum_fit_end <- suppressWarnings(summary(fit_end))
}
        > Number_Beta <- as.numeric(nrow(sum_fit_end$coefficients))
        > Decision <- 0
```

```
> i <- 1
> while (i <= Number_Beta) {
    > x <- ifelse(sum_fit_end$coefficients[i,4]<=Alpha,"Sig","NoSig")
    > Decision[i] <- x
    > i = i+1 }
> view <- data.frame(sum_fit_end$coefficients)
> view <- mutate(view, Decision )
> NoSig <- subset(view,view$Decision == "NoSig")
> RowNoSig <- as.numeric(nrow(NoSig))
> Sig <-  subset(view,view$Decision == "Sig")
> beta0 <- sum_fit_end$coefficients[1,4]
> if (beta0 <= Alpha && RowNoSig > 0) {
> delete.Intercept <- row.names(Sig)
> Newdata1 <- Newdata[delete.Intercept]
> while(RowNoSig > 0) {
> Newdata1 <- Newdata[delete.Intercept]
> Newdata1 <- data.frame(y,Newdata1)
> fit1 <- suppressWarnings(lm(y~0+., data=Newdata1))
> sumfit1 <- suppressWarnings(summary(fit1))
> Number_Beta1 <- as.numeric(nrow(sumfit1$coefficients))
> Decision1 <- 0
> i <- 1
> while( i <= Number_Beta1 ) {
```

```
        > x <- ifelse(sumfit1$coefficients[i,4]<= Alpha,"Sig","NoSig")
        > Decision1[i] <- x
        > i = i+1}
> view1 <- data.frame(sumfit1$coefficients)
> view1 <- mutate(view1, Decision1)
> Sig <- subset(view1,view1$Decision == "Sig")
> delete.Intercept <- row.names(Sig)
> delete.Intercept <- delete.Intercept[-1]
> NoSig <- subset(view1,view1$Decision == "NoSig")
> RowNoSig <- as.numeric(nrow(NoSig))
> RowNoSig = RowNoSig+0
}
> error <- fit1$residuals
> error.Group <- factor(error<=median(error))
> Normal <- shapiro.test(error)
> variance <- leveneTest(error,group = error.Group)
> variance_p <- variance$`Pr(>F)`[1]
> plot(fit1,1)
> plot(fit1,2)
> print(suppressWarnings(summary(fit1)))
> writeLines(c ("----------------------","Checking  Assumptions",  "----------
------"))
```

```
> ifelse(Normal$p.value <= Alpha,
print.noquote (" The errors do not follow a    normal distribution."),
print.noquote (" The errors follow a normal distribution."))

> ifelse( variance_p <= Alpha,
print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
print.noquote(" The variance of the errors is constant (Homoscedastic)."))

> list(

        coefficients = fit1$coefficients,

        residuals = fit1$residuals,

        fitted.values = fit1$fitted.values,

        rank = fit1$rank,

        df.residual = fit1$df.residual,

        call = fit1$call,

        terms = fit1$terms,

        model = fit1$model,

        lambda = lambda

)
} else if (beta0 <= Alpha && RowNoSig <= 0) {

> error_end <- fit_end$residuals

> error.Group <- factor(error_end<=median(error_end))

> Normal <- shapiro.test(error_end)

> variance <- leveneTest(error_end,group = error.Group)

> variance_p <- variance$`Pr(>F)`[1]

> plot(fit_end,1)
```

```
> plot(fit_end,2)

> print(suppressWarnings(summary(fit_end)))

> writeLines(c ("----------------------","Checking  Assumptions",  "----------------------"))

> ifelse(Normal$p.value <= Alpha,
print.noquote (" The errors do not follow a normal distribution."),
print.noquote (" The errors follow a normal distribution."))

> ifelse( variance_p <= Alpha,
print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
print.noquote(" The variance of the errors is constant (Homoscedastic)."))

> list(

        coefficients = fit_end$coefficients,

        residuals = fit_end$residuals,

        fitted.values = fit_end$fitted.values,

        rank = fit_end$rank,

        df.residual = fit_end$df.residual,

        call = fit_end$call,

        terms = fit_end$terms,

        model = fit_end$model,

        lambda = lambda

)

} else if (beta0 > Alpha && RowNoSig > 0) {

> delete.Intercept <- row.names(Sig)

> Newdata1 <- Newdata[delete.Intercept]
```

```
> while(RowNoSig > 0) {
> Newdata1 <- Newdata[delete.Intercept]
> Newdata1 <- data.frame(y,Newdata1)
> fit1 <- suppressWarnings(lm(y~0+.,data=Newdata1))
> sumfit1 <- suppressWarnings(summary(fit1))
> Number_Beta1 <- as.numeric(nrow(sumfit1$coefficients))
> Decision1 <- 0
> i <- 1
> while( i <= Number_Beta1 ) {
        > x <- ifelse(sumfit1$coefficients[i,4]<= Alpha,"Sig","NoSig")
        > Decision1[i] <- x
        > i = i+1}
> view1 <- data.frame(sumfit1$coefficients)
> view1 <- mutate(view1, Decision1)
> Sig <- subset(view1,view1$Decision == "Sig")
> delete.Intercept <- row.names(Sig)
> NoSig <- subset(view1,view1$Decision == "NoSig")
> RowNoSig <- as.numeric(nrow(NoSig))
> RowNoSig = RowNoSig+0
}
> error <- fit1$residuals
> error.Group <- factor(error<=median(error))
```

```
> Normal <- shapiro.test(error)

> variance <- leveneTest(error,group = error.Group)

> variance_p <- variance$`Pr(>F)`[1]

> plot(fit1,1)

> plot(fit1,2)

> print(suppressWarnings(summary(fit1)))

> writeLines(c ("----------------------","Checking  Assumptions","----------
----"))

> ifelse(Normal$p.value <= Alpha,
print.noquote (" The errors do not follow a normal distribution."),
print.noquote (" The errors follow a normal distribution."))

> ifelse(variance_p <= Alpha,
print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
print.noquote(" The variance of the errors is constant (Homoscedastic)."))

> list(

        coefficients = fit1$coefficients,

        residuals = fit1$residuals,

        fitted.values = fit1$fitted.values,

        rank = fit1$rank,

        df.residual = fit1$df.residual,

        call = fit1$call,

        terms = fit1$terms,

        model = fit1$model,

        lambda = lambda
```

```
        ) }
else {
        > delete.Intercept <- row.names(Sig)
        > Newdata1 <- Newdata[delete.Intercept]
        > Newdata1 <- data.frame(y,Newdata1)
        > fit1 <- suppressWarnings(lm(y~0+.,data=Newdata1))
        > sumfit1 <- suppressWarnings(summary(fit1))
        > error <- sumfit1$residuals
        > error.Group <- factor(error<=median(error))
        > Normal <- shapiro.test(error)
        > variance <- leveneTest(error,group = error.Group)
        > variance_p <- variance$`Pr(>F)`[1]
        > plot(fit1,1)
        > plot(fit1,2)
        > print(suppressWarnings(summary(fit1)))
        > writeLines(c ("----------------------","Checking  Assumptions","----------
        ----"))
        > ifelse(Normal$p.value <= Alpha,
        > print.noquote (" The errors do not follow a normal distribution."),
        > print.noquote (" The errors follow a normal distribution."))
        > ifelse( variance_p <= Alpha,
        print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
        print.noquote(" The variance of the errors is constant (Homoscedastic)."))
        > list(
```

```
                coefficients = fit1$coefficients,
                residuals = fit1$residuals,
                fitted.values = fit1$fitted.values,
                rank = fit1$rank,
                df.residual = fit1$df.residual,
                call = fit1$call,
                terms = fit1$terms,
                model = fit1$model,
                lambda = lambda
        ) } }
} else if (Normal$p.value >= Alpha && variance_p >= Alpha) {
        > writeLines(c ("-------------------------","Stepwise regression model ","-------
        ----------------"))
        > print(suppressWarnings(summary(fit1)))
        > plot(fit1,1)
        > plot(fit1,2)
        > writeLines(c("----------------------","Checking  Assumptions","----------------
        ----"))
        > writeLines ("[1] The errors follow a normal distribution.")
        > writeLines ("[1] The variance of the errors is constant (Homoscedastic).")
        > list(
                coefficients = fit1$coefficients,
                residuals = fit1$residuals,
                fitted.values = fit1$fitted.values,
```

```
                rank = fit1$rank,
                df.residual = fit1$df.residual,
                call = fit1$call,
                terms = fit1$terms,
                model = fit1$model
        ) }
} else  {
        > delete.Intercept <- row.names(Sig)
        > Newdata1 <- Newdata[delete.Intercept]
        > Newdata1 <- data.frame(y,Newdata1)
        > fit1 <- suppressWarnings(lm(y~0+.,data=Newdata1))
        > sumfit1 <- suppressWarnings(summary(fit1))
        > error <- sumfit1$residuals
        > error.Group <- factor(error<=median(error))
        > Normal <- shapiro.test(error)
        > variance <- leveneTest(error,group = error.Group)
        > variance_p <- variance$`Pr(>F)`[1]
> if (Normal$p.value <= Alpha || variance_p <= Alpha) {
        > Newdata1 <- data.frame(y,Newdata1)
        > Find.Lambda <- (boxcox(y~., data = Newdata1))
        > Find.Lambda2 <- data.frame(Find.Lambda$x,Find.Lambda$y)
        > Find.Lambda3 <-
        subset(Find.Lambda2,Find.Lambda$y==max(Find.Lambda$y))
```

```
> Optimal.lambda <- Find.Lambda3$Find.Lambda.x

> Optimal.lambda.true <- Find.Lambda3$Find.Lambda.x

> ifelse (Optimal.lambda >= 0.75 && Optimal.lambda < 1.5,Optimal.lambda
<- 1,

> Optimal.lambda <- Optimal.lambda)

> if (Optimal.lambda == 1 ) {

> writeLines(c("------------------------","Stepwise regression model ","----------
---------------"))

> print(suppressWarnings(summary(fit1)))

> writeLines(c("----------------------","Checking  Assumptions","----------------
----"))

> writeLines ("[1] The errors do not follow a normal distribution.")

> writeLines ("[1] The variance of the errors is not constant
(Heteroscedastic).")

> writeLines("------------------------\n Box cox transformation \n----------------
------")

> writeLines("Optimal lambda approximate to 1.")

> message("The multiple linear regression may not be appropriate for this
data.")

> plot(fit1,1)

> plot(fit1,2)

> list(

        coefficients = fit1$coefficients,

        residuals = fit1$residuals,

        fitted.values = fit1$fitted.values,
```

```
            rank = fit1$rank,
            df.residual = fit1$df.residual,
            call = fit1$call,
            terms = fit1$terms,
            model = fit1$model
        ) }
> if (Optimal.lambda != 1) {
        > writeLines(c("-------------------------------------------------","Regression model
        derived from data transformations","-------------------------------------------------
        "))
> if (Optimal.lambda> -0.25 && Optimal.lambda< 0.25){
        > lambda <- 0
        > writeLines(c("The lambda value utilized in the data conversion is 0","and
        transformation the dependent variable is in the form: y = log(y)"))
        > y.prime <- log(y)
        > Newdata1$y <- NULL
        > Newdata1 <- data.frame(y.prime,Newdata1)
        > fit_end <- suppressWarnings(lm(y.prime~0+.,data=Newdata1))
        > sum_fit_end <- suppressWarnings(summary(fit_end))
} else if (Optimal.lambda >= 0.25 && Optimal.lambda < 0.75 ) {
        > lambda <- 0.5
        > writeLines(c("The lambda value utilized in the data conversion is 0.5 ", "and
        > transformation the dependent variable is in the form: y=sqrt(y)"))
        > y.prime <- sqrt(y)
```

```
> Newdata1$y <- NULL

> Newdata1 <- data.frame(y.prime,Newdata1)

> fit_end <- suppressWarnings(lm(y.prime~0+.,data= Newdata1))

> sum_fit_end <- suppressWarnings(summary(fit_end))

} else if (Optimal.lambda >= 1.5 && Optimal.lambda <= 2 ) {

> lambda <- 2

> writeLines(c("The lambda value utilized in the data conversion is 2","and
transformation the dependent variable is in the form: y=y^2"))

> y.prime <- (y^2)

> Newdata1$y <- NULL

> Newdata1 <- data.frame(y.prime,Newdata1)

> fit_end <- suppressWarnings(lm(y.prime~0+.,data=Newdata1))

> sum_fit_end <- suppressWarnings(summary(fit_end))

} else if (Optimal.lambda <= -0.25 && Optimal.lambda > -0.75 ) {

> lambda <- -0.5

> writeLines(c("The lambda value utilized in the data conversion is -0.5","and
transformation the dependent variable is in the form: y=1/sqrt(y)"))

> y.prime <- 1/sqrt(y)

> Newdata1$y <- NULL

> Newdata1 <- data.frame(y.prime,Newdata1)

> fit_end <- suppressWarnings(lm(y.prime~0+.,data=Newdata1))

> sum_fit_end <- suppressWarnings(summary(fit_end))

} else if (Optimal.lambda <= -0.75 && Optimal.lambda > -1.5 ) {

> lambda <- -1
```

```
> writeLines(c("The lambda value utilized in the data conversion is -1","and t
ransformation the dependent variable is in the form: y=1/y"))
> y.prime <- 1/y
> Newdata1$y <- NULL
> Newdata1 <- data.frame(y.prime,Newdata1)
> fit_end <- suppressWarnings(lm(y.prime~0+.,data =  Newdata1))
> sum_fit_end <- suppressWarnings(summary(fit_end))
}else if (Optimal.lambda <= -1.5 && Optimal.lambda >= -2 ) {
> lambda <- -2
> writeLines(c("\n","The lambda value utilized in the data conversion is -2 ","
and transformation the dependent variable is in the form: y=1/y^2"))
> y.prime <- 1/(y^2)
> Newdata1$y <- NULL
> Newdata1 <- data.frame(y.prime,Newdata1)
> fit_end <- suppressWarnings(lm(y.prime~0+.,data =  Newdata1))
> sum_fit_end <- suppressWarnings(summary(fit_end))
}
> Number_Beta <- as.numeric(nrow(sum_fit_end$coefficients))
> Decision <- 0
> i <- 1
> while (i <= Number_Beta) {
    > x <- ifelse(sum_fit_end$coefficients[i,4]<=Alpha,"Sig","NoSig")
    > Decision[i] <- x
    > i = i+1}
```

```
> view <- data.frame(sum_fit_end$coefficients)
> view <- mutate(view, Decision )
> NoSig <- subset(view,view$Decision == "NoSig")
> RowNoSig <- as.numeric(nrow(NoSig))
> Sig <-  subset(view,view$Decision == "Sig")
> beta0 <- sum_fit_end$coefficients[1,4]
> if (beta0 <= Alpha && RowNoSig > 0) {
> delete.Intercept <- row.names(Sig)
> delete.Intercept <- delete.Intercept[-1]
> Newdata1 <- Newdata[delete.Intercept]
> while(RowNoSig > 0) {
> Newdata1 <- Newdata[delete.Intercept]
> Newdata1 <- data.frame(y,Newdata1)
> fit1 <- suppressWarnings(lm(y~., data=Newdata1))
> sumfit1 <- suppressWarnings(summary(fit1))
> Number_Beta1 <- as.numeric(nrow(sumfit1$coefficients))
> Decision1 <- 0
> i <- 1
> while( i <= Number_Beta1 ) {
        > x <- ifelse(sumfit1$coefficients[i,4]<= Alpha,"Sig","NoSig")
        > Decision1[i] <- x
        > i = i+1 }
> view1 <- data.frame(sumfit1$coefficients)
```

```
> view1 <- mutate(view1, Decision1)

> Sig <- subset(view1,view1$Decision == "Sig")

> delete.Intercept <- row.names(Sig)

> delete.Intercept <- delete.Intercept[-1]

> NoSig <- subset(view1,view1$Decision == "NoSig")

> RowNoSig <- as.numeric(nrow(NoSig))

> RowNoSig = RowNoSig+0 }

> error <- fit1$residuals

> error.Group <- factor(error<=median(error))

> Normal <- shapiro.test(error)

> variance <- leveneTest(error,group = error.Group)

> variance_p <- variance$`Pr(>F)`[1]

> plot(fit1,1)

> plot(fit1,2)

> print(suppressWarnings(summary(fit1)))

> writeLines(c ("----------------------",  "Checking  Assumptions","----------
------"))

> ifelse(Normal$p.value <= Alpha,
print.noquote (" The errors do not follow a normal distribution."),
print.noquote (" The errors follow a normal distribution."))

> ifelse( variance_p <= Alpha,
print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
print.noquote(" The variance of the errors is constant (Homoscedastic)."))

> list(
```

```
            coefficients = fit1$coefficients,
            residuals = fit1$residuals,
            fitted.values = fit1$fitted.values,
            rank = fit1$rank,
            df.residual = fit1$df.residual,
            call = fit1$call,
            terms = fit1$terms,
            model = fit1$model,
            lambda = lambda
        )
} else if (beta0 <= Alpha && RowNoSig <= 0) {
        > error_end <- fit_end$residuals
        > error.Group <- factor(error_end<=median(error_end))
        > Normal <- shapiro.test(error_end)
        > variance <- leveneTest(error_end,group = error.Group)
        > variance_p <- variance$`Pr(>F)`[1]
        > plot(fit_end,1)
        > plot(fit_end,2)
        > print(suppressWarnings(summary(fit_end)))
        > writeLines(c ("----------------------","Checking  Assumptions","----------
        ----"))
        > ifelse(Normal$p.value <= Alpha,
        print.noquote (" The errors do not follow a normal distribution."),
        print.noquote (" The errors follow a normal distribution."))
```

```
> ifelse( variance_p <= Alpha,
print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
print.noquote(" The variance of the errors is constant (Homoscedastic)."))
> list(
        coefficients = fit_end$coefficients,
        residuals = fit_end$residuals,
        fitted.values = fit_end$fitted.values,
        rank = fit_end$rank,
        df.residual = fit_end$df.residual,
        call = fit_end$call,
        terms = fit_end$terms,
        model = fit_end$model,
        lambda = lambda
)
} else if (beta0 > Alpha && RowNoSig > 0) {
> delete.Intercept <- row.names(Sig)
> Newdata1 <- Newdata[delete.Intercept]
> while(RowNoSig > 0) {
> Newdata1 <- Newdata[delete.Intercept]
> Newdata1 <- data.frame(y,Newdata1)
> fit1 <- suppressWarnings(lm(y~0+.,data=Newdata1))
> sumfit1 <- suppressWarnings(summary(fit1))
> Number_Beta1 <- as.numeric(nrow(sumfit1$coefficients))
> Decision1 <- 0
```

```
> i <- 1
> while( i <= Number_Beta1 ) {
        > x <- ifelse(sumfit1$coefficients[i,4]<= Alpha,"Sig","NoSig")
        > Decision1[i] <- x
        > i = i+1 }
> view1 <- data.frame(sumfit1$coefficients)
> view1 <- mutate(view1, Decision1)
> Sig <- subset(view1,view1$Decision == "Sig")
> delete.Intercept <- row.names(Sig)
> NoSig <- subset(view1,view1$Decision == "NoSig")
> RowNoSig <- as.numeric(nrow(NoSig))
> RowNoSig = RowNoSig+0 }
> error <- fit1$residuals
> error.Group <- factor(error<=median(error))
> Normal <- shapiro.test(error)
> variance <- leveneTest(error,group = error.Group)
> variance_p <- variance$`Pr(>F)`[1]
> plot(fit1,1)
> plot(fit1,2)
> print(suppressWarnings(summary(fit1)))
> writeLines(c ("----------------------","Checking  Assumptions","----------
----"))
```

```
> ifelse(Normal$p.value <= Alpha,
print.noquote (" The errors do not follow a normal distribution."),
print.noquote (" The errors follow a normal distribution."))

> ifelse( variance_p <= Alpha,
print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
print.noquote(" The variance of the errors is constant (Homoscedastic)."))

> list(
        coefficients = fit1$coefficients,
        residuals = fit1$residuals,
        fitted.values = fit1$fitted.values,
        rank = fit1$rank,
        df.residual = fit1$df.residual,
        call = fit1$call,
        terms = fit1$terms,
        model = fit1$model,
        lambda = lambda
)
} else {
> delete.Intercept <- row.names(Sig)
> Newdata1 <- Newdata[delete.Intercept]
> Newdata1 <- data.frame(y,Newdata1)
> fit1 <- suppressWarnings(lm(y~0+.,data=Newdata1))
> sumfit1 <- suppressWarnings(summary(fit1))
> error <- sumfit1$residuals
```

```
> error.Group <- factor(error<=median(error))

> Normal <- shapiro.test(error)

> variance <- leveneTest(error,group = error.Group)

> variance_p <- variance$`Pr(>F)`[1]

> plot(fit1,1)

> plot(fit1,2)

> print(suppressWarnings(summary(fit1)))

> writeLines(c ("----------------------","Checking  Assumptions","----------
----"))

> ifelse(Normal$p.value <= Alpha,
print.noquote (" The errors do not follow a normal distribution."),
print.noquote (" The errors follow a normal distribution."))

> ifelse( variance_p <= Alpha,
print.noquote(" The variance of the errors is not constant (Heteroscedastic)."),
print.noquote(" The variance of the errors is constant (Homoscedastic)."))

> list(

        coefficients = fit1$coefficients,

        residuals = fit1$residuals,

        fitted.values = fit1$fitted.values,

        rank = fit1$rank,

        df.residual = fit1$df.residual,

        call = fit1$call,

        terms = fit1$terms,

        model = fit1$model,
```

```
            lambda = lambda
        ) } }
} else if (Normal$p.value >= Alpha && variance_p >= Alpha) {
        > writeLines(c("-------------------------","Stepwise regression model","--
        ----------------------"))
        > print(suppressWarnings(summary(fit1)))
        > plot(fit1,1)
        > plot(fit1,2)
        > writeLines(c ("----------------------","Checking  Assumptions","----------------
        ----"))
        > writeLines("[1] The errors follow a normal distribution")
        > writeLines("[1] The variance of the errors is constant (Homoscedastic).")
        > list(
            coefficients = fit1$coefficients,
            residuals = fit1$residuals,
            fitted.values = fit1$fitted.values,
            rank = fit1$rank,
            df.residual = fit1$df.residual,
            call = fit1$call,
            terms = fit1$terms,
            model = fit1$model
        )
        } } }
}
```

VITAE

Name Miss Thidarat Thongsri

Student ID 6410220009

Educational Attainment

Degree	Name of Institution	Year of Graduation
Bachelor of Science (Statistics)	Prince of Songkla University	2021

Scholarship Awards during Enrolment

Graduate Fellowship (Bachelor – Master), Faculty of Science Research Fund, Prince of Songkla University.

List of Publication

1. Thongsri, T., & Samart, K. (2022). Composite Imputation Method for the Multiple Linear Regression with Missing at Random Data. Computer Science, 17(1), 51-62.
2. Thongsri, T., & Samart, K. (2023). Development of Imputation Methods for Missing Data in Multiple Linear Regression Analysis. Lobachevskii Journal of Mathematics, 9(44).
3. Thongsri, T., & Samart, K. mlrpro: Perform Stepwise Regression with Verifying Assumptions and Identifying Possible Box-Cox Transformation. The R Journal. (Submitted).