2014-2015

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
Τμήμα Οικονομικών Επιστημών

Constructing computational algorithm with the double-bootstrap method for the estimation of technical efficiency scores in two-stage semi-parametric DEA models and economic application

ΝΑΤΣΙΟΠΟΥΛΟΣ ΚΛΕΑΝΘΗΣ
ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΤΖΕΡΕΜΕΣ ΝΙΚΟΛΑΟΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΟΙΚΟΝΟΜΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
2014-2015

Περίληψη

Η παρούσα εργασία χωρίζεται σε δύο μέρη. Πρώτα, εφαρμόζει τους αλγορίθμους που πρότειναν αρχικά οι Simar και Wilson (2007) και κατασκευάζει μία πλήρη συναρτήσεις για κάθε έναν από αυτούς στη γλώσσα προγραμματισμού R. Το μοντέλο που χρησιμοποιούν αυτοί οι αλγόριθμοι είναι ένα μοντέλο DEA δύο σταδίων διπλής Bootstrap. Αρχικά αποτελείτε από δύο στάδια. Το πρώτο είναι αυτό στο οποίο πραγματοποιείται η DEA και το δεύτερο είναι η ανάλυση παλινδρόμησης. Το πρώτο στάδιο είναι αρκετά απλό αφού η μόνη εκτίμηση που κάνει είναι για την αποτελεσματικότητα ενώ το δεύτερο αποτελείτε είτε από μία είτε από δύο στάδια bootstrap. Στην περίπτωση του Algorithm #2 που χρησιμοποιεί διπλή bootstrap, στον πρώτο βρόχο υπολογίζει τη μεροληψία και της αποτελεσματικότητας και υπολογίζει ξανά τις διορθωμένες για μεροληψία αποτελεσματικότητες. Μετά συνεχίζει με τη δεύτερη bootstrap όπου κάποιες συνδιακυμάνσεις παλινδρομούνται πάνω στις εκτιμημένες διορθωμένες για μεροληψία αποτελεσματικότητες για να εξετάσουμε την επίδρασή τους πάνω σ' αυτές. Το δεύτερο μέρος είναι η οικονομική εφαρμογή που βασίζεται σ' αυτά τα μοντέλα. Η επίδραση που έχουν τα έξοδα διαφήμισης και επικοινωνίας στην αποτελεσματικότητα των τραπεζών στις Η.Π.Α. εξετάζεται και αποκαλύπτει κάποια ενδιαφέροντα αποτελέσματα. Τα τελικά αποτελέσματα υποδεικνύουν ότι μια τράπεζα θα πρέπει να συνεχίζει να ξοδεύει χρήματα στη διαφήμιση και τις επικοινωνίες μέχρι να φτάσει στο επίπεδο που η αναποτελεσματικότητά της θα αρχίσει να χρησιμοποιείτε για να ελαττώνεται.

Abstract

The present thesis has two parts. Firstly, it implements the algorithms originally suggested by Simar and Wilson (2007) and constructs a full functions for each one of them in the programming language R. The model that these algorithms use is a two-stage double bootstrap DEA. It consists primarily of two stages. The first one is where the DEA procedure takes place and the second one is the regression analysis. The first stage is quite simple as the only estimation is for the efficiency scores, while the second one consists of either one or two bootstrap stages. In the case of Algorithm #2 that makes use of a double-bootstrap,in the first loop it estimates the bias of the efficiency scores and computes again the bias-corrected efficiencies. Then it continues with the second bootstrap where some covariates are regressed on the estimated bias-corrected efficiencies in order to examine the effect they have on them. The second part is the economic application based on these models. The effect that the advertising and communications expense have upon the efficiency of the banks operating in U.S. is tested, yielding some interesting results. The final results indicate that a bank should keep spending money for advertising and communications until they reach the turning point where the bank's inefficiency will start to decline.

Key words: algorithms, efficiency, two-stage DEA, double-bootstrap, R, banks

Acknowledgements

I would like to thank my advisor Professor Nikolaos Tzeremes for the continuous support of my research, his guidance, patience, motivation and enthusiasm. I also owe special thanks for providing the data and for the valuable comments.

I would also like to thank the rest of my thesis committee Professor Ilias Kevork and Stephanos Papadamou as well as the rest of my Professors for their valuable help and knowledge along the years of my studies.

Last but not least, I would like to express my sincere gratitude to my family and friends for supporting me through all these years.

Contents

1. Introduction

1.1. Summary of Proposal

The present thesis has two primary goals. The first goal is to construct the two algorithms described by Simar and Wilson (2007) and the second one is the economic application using these algorithms.

The first algorithm uses a two-stage semi-parametric bootstrap model that firstly computes the efficiency scores using Data Envelopment Analysis (DEA) making use of some input and output variables and the second stage uses truncated regression to estimate the impact of some external variables upon these efficiencies.

The second algorithm uses a two-stage semi-parametric double bootstrap model that is exactly like the first algorithm; however, after the calculation of the efficiency scores, a bootstrap loop is used to re-estimate the efficiencies but this time removing the bias. The second stage, as in the first algorithm, regress some external variables upon the bias corrected efficiencies.

After the construction of the two algorithms, a data sample with various variables for Banks operating in U.S. is used for the economic application. The main subject here is to determine how the Advertising and Communications expense of the banks affect their efficiency.

1.2. Research Questions/ Goals

There are two main reasons why this specific research topic was chosen. The algorithms described by Simar and Wilson (2007) are quite popular around the literature and they have been used in many applications on the field of production economics (Halkos and Tzeremes, 2013), (Alexander et. al., 2010), (Keramidou and Mimis, 2011), (Latruffe et al., 2008), (Wijesiri et al., 2015), (Lee and Worthington, 2014). There are many programming languages and a great variety of packages that one can use to build these algorithms step by step but this can take time, depending on the complexity of the procedure. However, there is not yet an integrated package or a single function to do this, in spite of the great interest around the topic.

So, the first goal is the construction of a full integrated function for each algorithm provided by Simar and Wilson (2007), so everyone can make use of it and take advantage of the direct use instead of trying to build each step on his/her own. The algorithms are written in the programming language R, as it is an open source language and can be used from everyone without restrictions.

About the second part, the economic application, many studies have been done by researcher about the banks' performance (Jackson and Fethi, 2000), (Aly et al., 1990), (Hughes and Mester, 2008), (Weill, 2004). However, no one has study the impact of advertising and communications expense of the banks on their technical efficiency yet.

Thus, these two research questions are about to be answered here. How the advertising as well as the communication expenses of the American banks affects their technical efficiency.

2. Literature Review

2.1. Literature on Methods

2.1.1. DEA

The method we usually use to measure the performance and estimate the efficiency of different entities is called Data Envelopment Analysis (DEA). This is a mathematical programming technique that has been explicitly used for about 30 years now. Gattoufi et al. (2004) provide a very comprehensive list with the literature of DEA since these yeas.

In a production process, a firm tries to transform its inputs into outputs. Due to the increasingly competitive environment that the companies and the organizations in general operate, everyone has to be competitive enough to cope with their competitors. From economic perspective, there are two ways for an organization to be more efficient and thus more competitive. The first one is trying to minimize their cost (inputs) given a certain expected level of output. These cost minimization strategies are mostly followed by organizations in sectors that uncertainty plays a crucial role. A representative example is the agricultural sector where exogenous, like environmental factors, can definitely affect the production and at the same time these factors cannot be fully (or in some situations not at all) controlled by the organizations but they surely affect they inputs and outputs. The other strategy that can be followed is the one that the goal is to maximize the outcome (outputs) or more generally the effectiveness of the organization, given a certain level of inputs.

Each organization has to take a critical decision about the level of inputs they will use to reach a certain level of output or the way that they will handle their given inputs to reach the maximum level of outputs. In both strategies, the final goal is to maximize the efficiency. In a decision making problem like this, each organization is called a Decision Making Unit (DMU).

It's clear that the inputs are mostly under the control of the DMUs but there are some other environmental variables that also affect the production process. The difference here is that these variables usually affect the inputs and the outputs too, so the DMU has less control over the production process.

The most common way to measure efficiency is constructing a cost function made from the cost/effectiveness ratio resulted from the combination of many different input and output variables. As in most of the times is not easy to have the same measure for cost at every input variable or the same measure for effectiveness at every output variable, it is preferred to define the efficiency scores as the output/input ratio. This is what Charnes, Cooper and Rhodes used to measure the relative efficiency of a DMU. Two definitions of efficiency provided by Charnes, Cooper and Rhodes are given below:

Extended Pareto-Koopmans Efficiency: A fully efficient DMU is a DMU that cannot improve any of its inputs or outputs without worsening any of the other inputs or outputs.

Relative **Efficiency:** A DMU is fully efficient if and only if, based on the available information; there are no other DMUs that perform better showing that the focused one can improve its inputs or outputs without worsening any of it inputs or outputs.

The first DEA model was presented by Charnes, Cooper and Rhodes (CCR) in the 1970s following the work of Farrell (1957) who made the inception of DEA.

Except of the introduction of the DEA idea, Farrell also distinguished the different kinds of efficiencies. He separated "technical efficiency" from other kinds of efficiencies like "allocative", "scale" or "scope" efficiency.

There are many different models, many different definitions of what we mean by "efficiency", various assumptions and for each of those variants it takes lines of mathematical and graphical explanations for the reader to understand this huge topic. As the subject of this thesis is not the DEA models in particular and also for space saving reasons, the different models are not explained further here. The reader who is interested in more details can take a look at Cooper, Seiford and Zhu (2011).

The two basic DEA models are the so called CCR (Charnes et al. 1978) and BCC (Banker et al. 1984). For each of those models there are both input and output oriented versions. The difference between CCR and BCC model is that the first one allows Returns To Scale (RTS) to be only Constant Returns to Scale (CRS). The BCC model has one extra constraint that allows it to evaluate for Variable Returns to Scale (VRS) including increasing, constant and decreasing returns.

The boundary of the production possibility set P is called the technology or the production frontier. The firms that lie on this boundary are the technical efficient firms while those that operate in the interior of P, are the technical inefficient firms.

2.1.2. Two Stage DEA

The two-stage approach is widely popular in the literature of DEA as it has been used in many published studies and working papers. In a typical two-stage DEA model the efficiency scores are calculated in the first stage and then the estimated efficiencies are regressed on some covariates. Different variations of the two-stage DEA model also exist as in some studies, functions of the estimated efficiencies like ratios, Malmquist indices or other transformations (e.g. logarithmic) are regressed upon the covariates instead of the efficiency scores itself in the second stage.

Other variations suggest different regression methods like OLS, censored (Tobit) or truncated models.

The application of a censored model seems logical because of the nature of the DEA estimations of the efficiencies in the first stage. The argument behind this is that it tends to have many efficiency scores equal to one by construction. For many researchers this is a fact that makes these estimates to be censored whereas for some others this issue is not taken

into account and a simple OLS model is used. The problem that arises here is that if one assumes that the estimated efficiencies indeed tend to have many values equal to one, the results from the OLS are biased but in most of the studies, researchers take this property into account and instead use a censored (usually tobit) model (Aly et al., 1990), (Jackson and Fethi, 2000).

As Simar and Wilson (2007) report, another serious problem in the two-stage DEA models that in most of the cases arises is that the efficiency estimates are serially correlated. The reason why this happens comes from the fact that the existence of a DMU lying on the estimated frontier, affects the efficiency score of the other DMUs. This problem of serial correlation of estimated efficiencies from the DEA approach is logical as these estimates are based on the estimated frontier that is not the true unobserved one, as the maximum efficiency score is attached to the relatively most efficient DMU given a particular sample. This can also cause serial correlation in the estimated residuals from the OLS method in the second stage.

These are some of the most serious and usual problems in the context of the two-stage DEA models but they can be treated in one way or another. Another very important issue is the Data Generating Process (DGP) of the efficiencies estimated from the DEA. Because of the non-parametric maximum likelihood nature of the estimated frontier, is necessary to describe how the unobserved true frontier, and therefore the estimated one, is derived. Without specifying this DGP, any results coming from a two-stage DEA is meaningless. For this reason, Simar and Wilson (2007) described the first consistent DGP ever done for such models in order to make inference about the results. This DGP is given also here (1).

2.1.3. Bootstrap

The bootstrap is a nonparametric simulation method relevant to the parametric Monde Carlo. Therefore, one doesn't have to know the distribution of the population as in the parametric methods. The bootstrap is a very useful simulation tool based on the resampling method that is widely used to construct confidence intervals and calculate standard errors for the statistics of interest.

Usually, one uses an estimation of a statistic $\hat{\theta}$ to estimate the population parameter θ but in most of the cases the sampling distributions is unknown. In cases like this we can use the bootstrap distribution to approximate the actual sampling distribution.

The bootstrap procedure is very simple. We randomly draw n observations with replacement from the observed data set, where n is the number of observations that the initial observed data set includes. We keep doing many times (L times) this until we create L data set simulations and then for each one of them we calculate a statistic named bootstrap estimation $\hat{\theta}_i^*$ so that the distribution of the bootstrapped estimations simulated the sampling distribution. Next we can use these simulated estimates to define confidence intervals or calculate statistics like the mean and the standard deviation. More details about bootstrap and DEA are given in the next sections while the process of the analysis will go on.

2.2. Literature on Topic

2.2.1. Banks' Performance and Technical Efficiency

The crucial role that the banks play in the economy, especially when a crisis or difficult situations are faced, makes it a hot topic for research and further investigation. As banks are the greatest financial intermediations they differ from different types of firms. So, is not only about the impact that the banks' choices have upon the whole word, but the internal performance of the bank itself as an organization is very important. A potential bankruptcy of a bank due to bad performance is a fact that would have horrible consequences on the whole economy, as happened many times before in the history.

The usage of DEA to measure the efficiency of the banks is very common among the literature. One of the advantages is that it doesn't require the use of prices as other methods and only makes use of various inputs and outputs. This is indeed very useful as it is very hard to find prices for the inputs and outputs that a bank uses. There is a great debate about whether the choices of a bank affect its risk or not. In other words, are the banks operates under the manager's decisions or the owner's? The managers' decisions and their effect on the efficiency of the banks is something that is under focused investigation the recent years.

Many researchers have investigate the banking efficiency, some of these studies are (Jackson and Fethi, 2000), (Weill,), (Berger and Mester, 2003). To mention some of them, Tahir et al. (2009), studied the Malaysian banks' efficiency for the years 2000-2006 and found that for the years 2005-2006 banks have easier access to better technologies and different environmental and this fact makes them perform more efficient than the previous years. Hughes and Mester (2008) provide some simple explanations about the efficiency in banking, explaining the theory behind the bank's performance, the models that are used to measure their efficiency and some interesting results from empirical studies.

Although various kinds of efficiencies like allocative or scale efficiency are usually studied, our focus here is strictly on the technical efficiency of a bank and especially on the effect that some environmental variables that banks faces (advertising and communications expense) have on them.

3. Methodology

3.1. Theoretical and Statistical Approach

In this section are demonstrated the statistical model and the necessary assumptions for the models and the algorithms that follow.

Firstly, let's describe the variables that are used here. The input variables can be represented as a p-dimensional matrix $(n \times p)$ that takes non-negative real values as $x = (\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_p}) \in \mathbf{R_+^p}$ where each x_j $(j = 1, \dots, p)$ is a vector with n observations. The output variables can be represented as a q-dimensional matrix $(n \times q)$ that takes also non-negative real values as $y = (\mathbf{y_1}, \mathbf{y_2}, \dots, \mathbf{y_p}) \in \mathbf{R_+^q}$ where each y_j $(j = 1, \dots, q)$ is a vector

with n observations. The environmental variables can be described as an r-dimensional matrix $(n \times r)$ that takes real values $z = (z_1, z_2, \dots, z_r) \in R^r$ where each z_j $(j = 1, \dots, r)$ is a vector with n observations and has no further restrictions as the two previous vectors had. In addition, p, q and r denote the number of input, output and environmental variables respectively.

3.1.1. Theoretical Approach

In order to continue, it is necessary to set some basic but essential assumptions of the statistical model. The first assumption is about to define the semi-parametric DGP. The rest of the assumptions are about to describe in details the DGP.

This process is the one that generates the data in φ_n.The equation below is the one that describes the Data Generating Process (DGP) from which the observation set $\varphi_n = \{(x_i, y_i, z_i)\}_{i=1}^n$ is defined.

$$f(x_i, y_i, z_i) = f(x_i, \eta_i, \delta_i, z_i) = f(x_i, \eta_i | \delta_i, z_i) f(\delta_i | z_i) f(z_i) \quad (1)$$

In order to make this sequence understandable, (1) is interpreted here from right to left:

Consider the firm i facing the environmental variables z_i. These variables are drawn from $f(z_i)$.

Given these environmental variables, the firm faces an efficiency level δ_i. Thus, the efficiency of a firm is somehow based on some variables z_i. In other words, these environmental variables have an effect on the efficiency of the firm[1]. This efficiency δ_i is drawn from $f(\delta_i | z_i)$.

The input x_i as well as a part of the output variables, the angle η_i[2] that the firm faces is based on the environmental variables z_i and the efficiency level δ_i. Notice that while the firm has the control of its inputs, they are also restricted by some environmental variable and the efficiency level. On the other hand, the outputs are party under the control of the firm (let's say only the η_i part) and under some constrains (δ_i andz_i), as a part of y_i (the efficiency level δ_i) is anyway produced as the outcome of the environmental variables' impact. The x_i and η_i are drawn from $f(x_i, \eta_i | \delta_i, z_i)$.

This procedure results in $f(x_i, \eta_i, \delta_i, z_i)$ which is equivalent to $f(x_i, y_i, z_i)$ after the transformation of the polar coordinates (η_i, δ_i) to Cartesian coordinates (y_i).

The two-stage model presented here has to be statistically consistent in both the first and the second stage and also be consistent with the economic theory.

We can categorize the rest 8 assumptions in 3 main categories. The assumptions A1-A3 are used to ensure the properties of the environmental variables in the model (consistency of

[1] In our case this is the effect of advertising and communications expense on the efficiency of the banks.

[2] For an explicit explanation about the nature of η_i and δ_i, see the Appendix.

the second stage estimators). The assumptions A4-A6 cover the classical microeconomic theory assumptions while the assumptions A7-A8 are used to assure the consistency of the DEA estimators that appear in the first stage. These 8 assumptions are presented below in more details:

A1. The input $\boldsymbol{x}$, output $\boldsymbol{y}$, and environmental $\boldsymbol{z}$ variables are identically, independently distributed (iid) random variables with probability density function (pdf) $\boldsymbol{f(x,y,z)}$.
$\boldsymbol{P = \{(x,y)|x\ can\ produce\ y\}}$ is the production set where $\boldsymbol{P \subset R_+^{p+q}}$.
Moreover, we can say that $\boldsymbol{f(x,y,z)}$ has support over $\boldsymbol{P \times R^r}$.

The overall meaning of this assumption is that the production set $\boldsymbol{P}$, and therefore it's boundary, is defined only by the inputs and outputs ($\boldsymbol{P \subset R_+^{p+q}}$). though the last line explains that the environmental variables have an impact on $\boldsymbol{P}$ too ($\boldsymbol{P \times R^r}$), but in a different way that is explained in the next assumption.

A2. The way that the efficiency level $\boldsymbol{\delta_i}$ is created given the environmental variables $\boldsymbol{z_i}$, meaning the generation of $\boldsymbol{f(\delta_i|z_i)}$ in (1) is given by the following mechanism.

$$\boldsymbol{\delta_i = \psi(z_i,\beta) + \varepsilon_i \geq 1} \tag{2}$$

In this formation, $\boldsymbol{\psi}$ is a smooth, continuous function, $\boldsymbol{\beta}$ is the vector of parameters and $\boldsymbol{\varepsilon_i}$ is the error term which is an identically, independently distributed (iid) random variable independent of $\boldsymbol{z_i}$.

The assumption simply explains the way that $\boldsymbol{z_i}$ interacts with $\boldsymbol{\delta_i}$, and given the function $\boldsymbol{\psi}$ it is practically implemented with the regression in the second-stage. It also makes clear that the way that $\boldsymbol{z_i}$ affects $\boldsymbol{P}$ has nothing to do with the Production Possibility Function (PPF) that is represented by the frontier of $\boldsymbol{P}$ as **A1** indicates, but the only effect it has is upon the efficiency score $\boldsymbol{\delta_i}$ of each DMU.

Together, assumptions **A1** and **A2** imply to a separability condition about the production set $\boldsymbol{P}$ and the effect of $\boldsymbol{z}$ on $\boldsymbol{\delta}$[3] and so the dependence between $\boldsymbol{y}$ and $\boldsymbol{z}$ that arises by the DGP in (1).

[3] Alternative formulations about the production possibility set can be made including an extra constraint about z and take it into account in the first-stage of linear programming (DEA). In this case, z influences the boundary surface in contrast with our assumption where it affects only the mean and variance of the efficiencies. In addition, in the case that z influences the boundary surface there is no reason to proceed in the second-stage. Model like this are discussed by Coelli et al. (1998).

A3. The error term ε_i in (2) is distributed $N(0, \sigma_\varepsilon^2)$ with each i element of ε_i to be selected with left truncation at $1 - \psi(z_i, \beta)$.

In this assumption two choices are made. The first one is about the distribution of ε_i and the second one is about the form of $\psi(\cdot)$. A full parametric method for truncated regression is assumed here as both of the two elements mentioned before are specified but other forms are possible too[4].

A4. $x(y) = \{x|(x, y) \in \mathrm{P}\}$ and $y(x) = \{y|(x, y) \in \mathrm{P}\}$ are closed and convex for every $y \in R_+^q$, and bounded for every $x \in R_+^p$ respectively. Thus, P is also closed and convex as the consideration for both $x(y)$ and $y(x)$ implies the knowledge of P and vice versa.

This assumptions simply explains that P is defined by the combinations of both x and y, (x, y) while at the same time the inputs of a firm are restricted by its outputs and vice versa. This is the assumptions about P as described by Shephard (1970) and Fare (1988).

A5. $(x, y) \notin P$ if $x = 0, y \geq 0$

This assumption means that there are no "free lunches", so there cannot be any level of production without the use of some inputs.

A6. If $(x, y) \in P$, then $(\bar{x}, y) \in P$ and $(x, \bar{y}) \in P$, where $\bar{x} \geq x$ and $\bar{y} \leq y$.

This is the "free disposability" assumption and means that the inputs and outputs are strongly disposable. So that a firm may not fully utilizes its inputs. Equivalently in technical terms it is an assumption of monotonicity of the technology.

A7. $f(x, y|z)$ is strictly positive and continuous in the interior of P for all z, for all $(x, y) \in P$ such that $(\theta^{-1}x, y) \notin P$ and $(x, \theta y) \notin P$ for $\theta > 1$.

This assumption implies the consistency of the estimator of P. One could give a free description of the meaning of this assumption, as any point (x, y) that is weighted as described above and so lies outside the frontier, does not belong to P. This, as well as the following assumption are based on Kneip et al. (1998) but adjusted to cover the environmental variables z.

A8. $\delta(x, y|P)$ is differentiable in both x and y for all (x, y).

---

[4] Leave the distribution of ε_i unspecified and employ the semi-parametric form of truncated regression (Honoré and Powell, 1994) or leave the function $\psi(\cdot)$ unspecified too and employ the non-parametric method for truncated regression (Lewbel and Linton, 2002).

This is an assumption about the smoothness of the frontier and it indicates that the estimator of δ_i is consistent if the probability to observe a firm in the neighborhood of any point on the frontier is equal to 1 as the sample increases in size. So the boundary line to be smooth, continuous and thus differentiable in any point[5].

3.1.2. Statistical Approach

Using the observation set φ_n, one can estimate the production set P with the following estimator $\hat{P}$:

$$\hat{P} = \{(x,y)|y \leq Yq, x \geq Xq, x'q = 1, q \in R_+^n \quad (3)$$

Where q denotes an $(n \times 1)$ vector of intensity variables, i is the unit vector $(n \times 1)$ and the capital letters X and Y denote the observation sets of x and y equivalently. It has been proved by Korostelev et al. (1995) that $\hat{P}$ is a consistent estimator of P under assumptions A1 – A8.

We define the measure of output technical efficiency δ_i for a particular point (DMU) $(x_i, y_i) \in R_+^{p+q}$ as following (Farrell, 1957):

$$\delta_i = \delta(x_i, y_i|P) \equiv sup\{\delta|(x_i, \delta y_i) \in P, \delta > 0\} \quad (4)$$

Where for every $(x_i, y_i) \in P$, there is a function $\delta(x_i, y_i|P) > 1$ that measures the Euclidean distance from this point to the boundary of P. Referring to the output oriented DEA, the direction of measurement (technical efficiency) is parallel to the output axes and orthogonal to the input axes, while referring to the input oriented model the direction of measurement is reversed.

Replacing the equation (3) in (a1) one can construct the Farrell efficiency measure estimator $\hat{\delta}$ in terms of linear programming as:

$$\hat{\delta}_i = \delta(x_i, y_i|\hat{P}) = \max\{\theta > 0|\theta y_i \leq Y_q, x_i \geq X_q, i'q = 1. q \in R_+^n\} \quad (5)$$

which is the empirical expression of (4).

A problem that appears with the non-parametric methods is the low rate of convergence that causes problems as is explained later. In this case, $\hat{\delta}_i$ has a very low rate of convergence as typically the non-parametric estimators have.

$$\hat{\delta}_i = \delta_i + O_p(n^{-2/(p+q+1)}) \quad (6)$$

[5] This is a stronger assumption than is typically required as Kneip et al. (1998) require only Lipschitz continuity for the distance function.

This rate of convergence decreases as n increases ($O_p(0)\ as\ n \to \infty$) and slows as $p+q$ increases ($O_p(1)\ as\ p+q \to \infty$). The second one is known as the "curse of dimensionality" that makes the convergence very slow when many input or output variables are used.

The previous explanation is the statistical context behind the estimation of the efficiency estimators $\hat{\delta}_i$ using DEA in the step [1] in both Algorithm #1 and Algorithm #2.

One can estimates β in (2) with maximum likelihood specifying the function $\psi(\cdot)$. A typical form for this function is simply $\psi(z_i, \beta) = z_i\beta$. This makes (2) to take the following specific form:

$$\delta_i = z_i\beta + \varepsilon_i \geq 1 \tag{7}$$

As the theoretical efficiency δ_i is unknown, one computes an estimation of this using DEA. This can be simply done using the observed inputs and outputs (x_i, y_i) in φ_n in (3) and (5) yielding $\hat{\delta}_i$. Then, in the second-stage one can regress the observed z_i on the estimated $\hat{\delta}_i$ as shown in (8).

$$\hat{\delta}_i = z_i\beta + \xi_i \geq 1 \tag{8}$$

This regression will lead to the estimation of $\hat{\beta}$ and the residuals $\hat{\xi}_i$. This is the statistical context behind the estimation of the coefficient estimators β using truncated regression in the step [2] in both Algorithm #1 and Algorithm #2.

The kind of regression that is used here for the estimation of (7) is the truncated regression for reasons that are described in (Simar and Wilson, 2007). In most of the cases before Simar and Wilson present this model, the regression method that were used was a censored (Tobit) model or sometimes a simple OLS. In the paper mentioned above, Simar and Wilson showed how faulty was the use of a Tobit model and compares it with the results from a truncated regression model. The (Figure 1) below shows clearly the superiority of the truncated regression over the Tobit models. One can see that the confidence intervals from the Tobit model are far beyond the vertical dashed line (the true parameter), while the confidence intervals form the truncated regression almost perfectly cover the true parameter, especially in the case of $p = q = 1$. Adding more dimentions arises the problem in the first-stage mentioned before in (6) because of the "course of dimensionality".

Figure 1
Source: Simar and Wilson (2007)

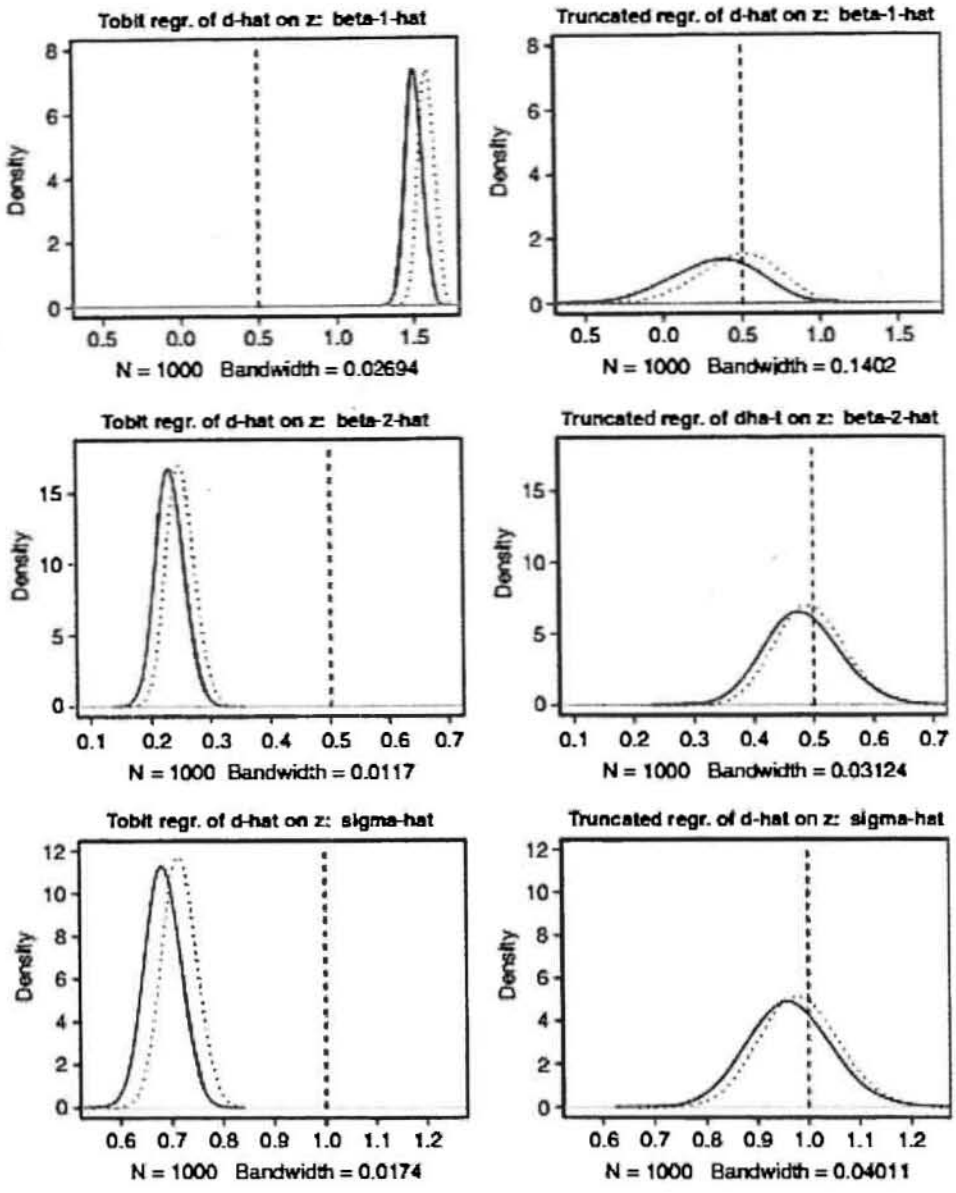


Fig. 1. Estimates of sampling densities, Algorithm #1 ($n = 400$; dotted curve for $p = q = 1$, solid curve for $p = q = 2$).

The problem that arises here is that the estimated efficiency scores δ_i are serially correlated by construction as DEA evaluates each DMU depending on the values that have been given to the other DMUs, in particular to the most efficient ones. As we can see in (5), δ_i depends on $\hat{P}$ and $\hat{P}$ depends on (x_i, y_i) as it's clear from (3), so consequently δ_i depends on (x_i, y_i). This serial correlation of δ_i passes on the estimated residuals ξ_i. Now recall that due to assumption **A2**, x and y are correlated with z. In most of the cases it's true that $f(x, y|z) \neq f(x, y)$. This means that the z variables are not independent with respect to

(x, y) and this is the reason why we apply the second stage regression. Interpreting this sequence one can see tha z is correlated with the residuals ξ_i.

Two problems arise here, the first one is the serial correlation in ξ_i and the second one is the correlation between ξ_i and z. The maximum likelihood estimators of β are still consistent but the problem disappears asymptotically with the rate of convergence of δ_i in (6) which is very slow, rather than the usual parametric convergence rate of $n^{-1/4}$. The serial correlation problem disappears also asymptotically with the same convergence rate, but for models where $p + q > 3$ in the first-stage, this rate is very slow for standard approaches as Simar and Wilson (2007) report (based on the inversion of log-likelihood negative Hessian). Another problem is that δ_i is also biased downward by construction. After the estimations of (8), the Algorithm #1 proceeds to the steps [3] and [4]. Except the usefulness of the bootstrap in the second stage, the bootstrap procedure is also used in the first stage as this is the only way to approximately reach the asymptotic distribution of δ_i in multivariate settings with $p + q > 2$ because of the complexity of these cases. Nevertheless, Gijbels et al. (1999) proved the asymptotic distribution of the output-oriented distance function (it can be also extended to the input distance function case) of Shephard (1970) for the case of $p = q = 1$ but this is not enough for the most applications. In contrast with Algorithm #1, Algorithm #2 takes care of this problem following the next procedure.

We can express δ_i as:

$$\delta_i = E(\delta_i) + u_i \tag{9}$$

Where $E(\delta_i) = 0$ and u_i is iid with zero mean. In turn, because δ_i is biased downward as mentioned before we have:

$$E(\delta_i) = \delta_i + BIAS(\delta_i) \tag{10}$$

Solving (10) for $BIAS(\delta_i)$ we have:

$$BIAS(\delta_i) = E(\delta_i) - \delta_i \tag{11}$$

So the true bias of δ_i is the expected value of δ_i (the estimated mean) minus the the true population mean.

Substituting for $E(\delta_i)$from (9) in (11) and solving for δ_i:

$$\delta_i = \delta_i - u_i - BIAS(\delta_i) \tag{12}$$

The equation in (12) is the analytical expression of (7). Substituting the right-hand side of (7) in (12) we have:

$$z_i\beta + \varepsilon_i = \delta_i - u_i - BIAS(\widehat{\delta}_i) \tag{13}$$

Solving (13) for δ_i we have another expression of (8) as:

$$\delta_i = z_i\beta + \varepsilon_i + u_i + BIAS(\widehat{\delta}_i) \tag{14}$$

Linking (14) with (8), one can see that the error term in (8) that causes the problems mentioned previously is:

$$\xi_i = \varepsilon_i + u_i + BIAS(\delta_i) \tag{15}$$

where ε_i is the theoretical error that follows all the assumption about the error terms and u_i has a zero mean and diminish asymptotically. So the problems contained in ξ_i are mostly inside the $BIAS(\delta_i)$. Thus, the problem can be easily solved just subtracting the bias from the estimated efficiencies $\bar{\delta}_i$.

The calculation of the bias is very easy using bootstrap methods. The first step to do this is to compute the bootstrap bias estimate of $BIAS(\delta_i)$:

$$\widehat{BIAS}(\hat{\delta}_i) = BIAS(\delta_i) + v_i \tag{16}$$

Where, as the sample size increases, the residual v_i has smaller magnitude than $BIAS(\bar{\delta}_i)$ and its variance becomes asymptotically negligible.

Expanding (11) we have:

$$BIAS(\bar{\delta}_i) = E(\hat{\delta}_i - \delta_i) \tag{17}$$

Following this, the bootstrap bias estimate $\widehat{BIAS}(\bar{\delta}_i)$ is:

$$\widehat{BIAS}(\hat{\delta}_i) = E(\delta_i^* - \bar{\delta}_i) = E(\delta_i^*) - \bar{\delta}_i \tag{18}$$

Where $E(\delta_i^*)$ is the mean of the all the bootstrap values and $\bar{\delta}_i$ represents the initial uncorrected estimation of the efficiency.

Taking advantage of the bootstrap technique where the initial estimator $\bar{\delta}_i$ plays the role of the population statistic and $E(\delta_i^*)$ represents the sample mean executed from the bootstrap sample distribution, one can go from (17) to (18).

Now the bias-corrected estimator of δ_i is:

$$\hat{\hat{\delta}}_i = \bar{\delta}_i - \widehat{BIAS}(\bar{\delta}_i) \tag{19}$$

So we finally have:

$$\hat{\hat{\delta}}_i + v_i - u_i = z_i\beta + \varepsilon_i \geq 1 \tag{20}$$

$$\hat{\hat{\delta}}_i \approx z_i\beta + \varepsilon_i > 1 \tag{21}$$

Where $\hat{\hat{\delta}}_i$ is a consistent estimate.

The steps [3] and [4] in Algorithm #2 follow this method, and regress the covariates on the bias-corrected estimate $\hat{\hat{\delta}}_i$.

The last part refers to the confidence intervals estimation. Two methods for confidence interval are described here, although other more sophisticate methods could be used for better results. The first method is the basic confidence interval. The estimated interval for this method is:

$$[2\hat{\beta} - q^{(\beta)}_{1-\alpha/2},\quad 2\hat{\beta} - q^{(\beta)}_{\alpha/2}] \tag{22}$$

Where $\hat{\beta}$ is the initial estimate, $q^{(\beta)}_{1-\alpha/2}$ is the upper quantile from the sorted bootstrap distribution and $q^{(\beta)}_{\alpha/2}$ the lower quantile from the sorted bootstrap distribution.

The other method is the percentile method. The percentile confidence intervals are constructed as:

$$[q^{(\beta)}_{\alpha/2},\quad q^{(\beta)}_{1-\alpha/2}] \tag{23}$$

3.2. Data

The data that were used for the empirical economic application part were received from the database of CompuStat. The data that finally used include 148 banks operating in U.S. covering the years 2006-2013 reaching the total number of observations ($n = 1184$). The variables that were used in the first stage to estimate technical efficiency include three input variables (Total Assets, Total Deposits and Employees) ($p = 3$) and two output variables (Total Securities and Gross Loans) ($q = 2$). For the second stage, two covariates were used to explain the estimated inefficiencies from the first-stage (Advertising Expense and Communications Expense) ($r = 2$) but various combinations of the logs of these two variables were also used to account for non-linearities. All variables except the number of Employees are measured in U.S. dollars. The choice of these variables as inputs and outputs was maid based on the economic theory and also on other empirical studies that use the same variables (Berger and Mester, 2003).

The descriptive statistics of the final balanced panel sample are presented below:

Table 1

	Inputs		
Statistic	Total Assets	Total Deposits	Employees
Min.	122.5	108.8	0.039
Q1	772.6	591.5	0.1827
Median	1966.8	1484.6	0.429
Mean	57777.7	33926.6	8.2975
Q3	6639.5	5128.4	1.4318
Max.	2415689	1287765	288
St. Dev.	265137.5	141433.5	35.6809

Table 2

Outputs		
Statistic	Total Securities	Gross Loans
Min.	3.9	93.4
Q1	132.8	501.1
Median	346.5	1294.8
Mean	9386.6	26903.9
Q3	1295.4	4308.5
Max.	507255	918498
St. Dev.	44899.97	112640.2

Table 3

Environmental		
Statistic	Advertising Expense	Communications Expense
Min.	0.017	0
Q1	0.523	0
Median	1.587	0.645
Mean	52.254	66.343
Q3	4.859	3.491
Max.	3143	5425
St. Dev.	269.774	414.696

The full data before the sampling was an unbalanced panel with a total of 7950 observations, containing 1117 different banks covering various years from 1999 to 2014. This full data unbalanced panel had too many missing values between the variables, so a careful cleaning in order to choose the best sample was necessary.

Note that in order to take the log of the variable Communications Expense, the value of 1 was arbitrary added to every observation of the variables as it would be impossible to take the log of 0. This is very common technique for situations like this (Puig-Junoy, 1998).

3.3. Cleaning and Selection Procedures

In the research process, in order to do any kind of analysis one usually needs data. In the real world, except the fact that good data is hard to be found even if they already exist in most of the times they are difficult to be read, we call this kind of data "raw data". In order to have easy access to the data first it needs to clean the raw data and prepare them for the analysis. The new easy manageable format of the data is called "tidy data".

The procedure of converting the raw data into tidy data is very often a time consuming and rigorous part of the research. For this reason here it is also provided, for educational reasons, the R code with which the raw data used in this thesis were cleaned.

The initial data file is a Microsoft Excel (.xls) file named "banks". Converting this file into a .Comma Separated Value (.csv) file, it is much easier to enter the data in R and work with them. So, the file "banks_csv" contains exactly the same information, the only difference is that this is a .csv file.

In this section are described the two R codes that are necessary before the rest of the analysis. The first code has the name "max_info_clean_data" and is used to clean the data and find the best balanced sample to use. The second one has the name "create_balanced_panel_data" and is used to create the balanced panel data after the choice of the sample.

It is essential to set the working directory of R in the folder where the data and the R files are, before running any of the following codes. One can do this simply writing the path of the folder in the command setwd(" ").
For example: setwd("C:/User/Desktop")

max_info_clean_data

This code firstly reads the data (max_info_clean_data, 1-2) and then keeps only the banks that operate in U.S. (max_info_clean_data, 3-7). Then, it saves in the object all_firms the name of each bank (in fact, the ticker that represents each bank), the first and the last year where this bank has recorded data (max_info_clean_data, 10-18).

In (max_info_clean_data, 23-87), all the possible combinations of year between 1999 and 2014 are tested to find which the best years to include in the sample are. In this procedure, the object $bounds$ that is defined in (max_info_clean_data, 29) ensures that each bank has data for, at least, the years that are tested each time. In (max_info_clean_data, 34-51) two more filters are used to ensure that the banks that have at least the required years, they also don't overcome these years, so the panel will be balanced. Then, the data are stored in the object $full\_data$ (max_info_clean_data, 52-71) and the final information for each combination is saved in the object $Total$ (max_info_clean_data, 78-83). In the first column of $Total$ is written the first year of this particular year range that is tested each time, in the second column the last year, in the third column the number of banks that these years include and in the fourth column the total number of observations for these years (years times number of banks, as the panel is balanced). Finally, the object $Total$ with all the possible combinations is sorted, with respect to the total number of observations, and the results are printed on the screen as well as presented in a table (max_info_clean_data, 88-92).

create_balanced_panel_data

This code simply creates the balance panel of our choice. The first few steps (create_balanced_panel_data, 1-20) are the same as in the previous code.

Here the user has to choose the preferable years and set the starting year in the object *lower* and the ending year in the object *upper* (create_balanced_panel_data, 24-25). In our case, the chosen years are 2006-2013. The steps (create_balanced_panel_data, 26-70) are the same steps that max_info_clean_data uses to construct the object $full\_data$ that contains the final balanced data set for the requested years.

In (create_balanced_panel_data, 72-76) the final data are saved in a txt file named "balanced_panel_data" and is automatically placed in the current working directory.

3.4. Algorithms

The first part of this thesis is the construction of the two algorithms initially described by Simar and Wilson (2007). Although the two algorithms presented here implement those that was previously mentioned, as this is one of the actual goals of this thesis, these algorithms can also been generally used to estimate various two-stage semi-parametric DEA models with a single or double bootstrap.

The theoretical procedure and steps along with the mathematical expressions and the corresponding R code interpretation of the two algorithms (Algorithm_1_simplified and Algorithm_2_simplified) described in Simar and Wilson (2007) is given below while the exact R code is given in the Appendix.

Algorithm_1_simplified and Algorithm_2_simplified are the simplistic versions of Algorithm_1 and Algorithm_2 respectively. This is because the structures of the last two algorithms are too complicated as they take account of every possible combination that the user can ask from the algorithms. Thus, Algorithm_1_simplified and Algorithm_2_simplified are presented here for educational reasons. The full and the simplified algorithms work exactly the same but the simplified versions result only in the confidence intervals of $\boldsymbol{\beta}$ and $\boldsymbol{\sigma_\varepsilon}$.

Algorithm #1

[1] Compute $\bar{\boldsymbol{\delta}}_{\boldsymbol{i}} = \bar{\boldsymbol{\delta}}(\boldsymbol{x}, \boldsymbol{y}|\bar{\boldsymbol{P}})\ \forall \boldsymbol{i} = \boldsymbol{1}, \dots, \boldsymbol{n}$ with (5) using the data in $\boldsymbol{\varphi_n}$.
(Algorithm_1_simplified, 25-26)

[2] Estimate the maximum likelihood estimators $\bar{\boldsymbol{\beta}}$ and $\bar{\sigma}_{\xi}$ with the truncated regression (8) of $\bar{\boldsymbol{\delta}}_{\boldsymbol{i}}$ on $\boldsymbol{z}$ using the $\boldsymbol{m} < \boldsymbol{n}$ observations where $\bar{\boldsymbol{\delta}}_{\boldsymbol{i}} > \boldsymbol{1}$.
(Algorithm_1_simplified, 33-47)

[3] Repeat the steps [3.1] – [3.3] $\boldsymbol{L}$ times to obtain the bootstrap set of estimates $\boldsymbol{A} = \{(\widehat{\boldsymbol{\beta}}^*, \widehat{\sigma}^*_{\xi})_b\}_{b=1}^{L}$.
(Algorithm_1_simplified, 59-100)

[3.1] For each $i = 1, \dots, m$, draw u_i from the normal distribution $N(0, \bar{\sigma}_\xi^2)$ with left truncation at $c = (1 - z_i\beta)$.
(Algorithm_1_simplified, 59-73)

[3.2] For each $i = 1, \dots, m$, compute $\delta_i^* = z_i\bar{\beta} + u_i$.
(Algorithm_1_simplified, 76)

[3.3] Estimate the maximum likelihood estimates $(\beta^*, \bar{\sigma}_\varepsilon^*)$ with the truncated regression of δ_i^* on z using the observations where $\delta_i^* > 1$.
(Algorithm_1_simplified, 79-93)

[4] Construct estimated confidence intervals for β and σ_ε.
(Algorithm_1_simplified, 115-152)

Algorithm #2

[1] Compute $\bar{\delta}_i = \bar{\delta}(x, y|\bar{P})\ \forall i = 1, \dots, n$ with (5) using the data in φ_n.
(Algorithm_2_simplified, 25-26)

[2] Estimate the maximum likelihood estimators $\bar{\beta}$ and $\bar{\sigma}_\xi$ with the truncated regression (8) of $\bar{\delta}_i$ on z using the $m < n$ observations where $\bar{\delta}_i > 1$.
(Algorithm_2_simplified, 33-47)

[3] Repeat the steps [3.1] – [3.4] L_1 times to obtain the bootstrap set of estimates $B = \{\widehat{\delta}^*_{i,b}\}_{b=1}^{L_1}$.
(Algorithm_2_simplified, 56-86)

[3.1] For each $i = 1, \dots, n$, draw u_i from the normal distribution $N(0, \bar{\sigma}_\xi^2)$ with left truncation at $c = (1 - z_i\beta)$.
(Algorithm_2_simplified, 56-70)

[3.2] For each $i = 1, \dots, n$, compute $\delta_i^* = z_{i,j}\bar{\beta}_j + u_i$.
(Algorithm_2_simplified, 73)

[3.3] For all $i = 1, \dots, n$, construct the new input and output variables $x^* = x$ and $y^* = y\bar{\delta}_i/\delta_i^*$.
(Algorithm_2_simplified, 76-77)

[3.4] Compute $\bar{\delta}^*_i = \widehat{\delta}(x_{i,j}, y_{i,j}|\widehat{P}^*)\ \forall i = 1, \dots, n$ with (5) replacing x and y with x^* and y^* respectively.
(Algorithm_2_simplified, 80-81)

[4] For each $i = 1, \dots, n$, compute the bias-corrected estimator $\widehat{\bar{\delta}}_i$ using the initial estimate $\bar{\delta}_i$ and the bootstrap estimates in B.
(Algorithm_2_simplified, 90-95)

[5] Estimate the maximum likelihood estimators $\widehat{\widehat{\boldsymbol{\beta}}}$ and $\widehat{\widehat{\sigma}}_{\varepsilon}$ with the truncated regression (21) of $\widehat{\widehat{\boldsymbol{\delta}}}_i$ on $\boldsymbol{z}$ using the $\boldsymbol{m} < \boldsymbol{n}$ observations where $\widehat{\widehat{\boldsymbol{\delta}}}_i > \mathbf{1}$.
(Algorithm_2_simplified, 100-114)

[6] Repeat the steps [6.1] – [6.3] $\boldsymbol{L_2}$ times to obtain the bootstrap set of estimates $\boldsymbol{C} = \{(\widehat{\boldsymbol{\beta}}^*, \widehat{\sigma}_{\varepsilon}^*)_b\}_{b=1}^{L_2}$ where $\boldsymbol{k} = \mathbf{1}, \dots, \boldsymbol{r} + \mathbf{2}$.
(Algorithm_2_simplified, 124-165)

> [6.1] For each $\boldsymbol{i} = \mathbf{1}, \dots, \boldsymbol{n}$, draw $\boldsymbol{u_i}$ from the normal distribution $\boldsymbol{N}(\mathbf{0}, \widehat{\widehat{\sigma}}_{\varepsilon}^2)$ with left truncation at $\boldsymbol{c} = (\mathbf{1} - \boldsymbol{z_i}\widehat{\boldsymbol{\beta}})$.
> (Algorithm_2_simplified, 124-138)
>
> [6.2] For each $\boldsymbol{i} = \mathbf{1}, \dots, \boldsymbol{n}$, compute $\widehat{\boldsymbol{o}}_i^{**} = \boldsymbol{z_i}\widehat{\widehat{\boldsymbol{\beta}}} + \boldsymbol{u_i}$.
> (Algorithm_2_simplified, 141)
>
> [6.3] Estimate the maximum likelihood estimates $(\widehat{\widehat{\boldsymbol{\beta}}}_j^*, \widehat{\widehat{\sigma}}_{\varepsilon}^*)$ with the truncated regression of $\boldsymbol{\delta}_i^{**}$ on $\boldsymbol{z}$ using the observations where $\boldsymbol{\delta}_i^{**} > \mathbf{1}$.
> (Algorithm_2_simplified, 144-158)

[7] Construct estimated confidence intervals for $\boldsymbol{\beta}$ and $\boldsymbol{\sigma_{\varepsilon}}$.
(Algorithm_2_simplified, 180-217)

These algorithmic steps are also presented here with the corresponding R code and their interpretation.

In Algorithm_1_simplified, one can give a set of input and another one of output variables. Each of these variable sets must be in one matrix (let's name $\boldsymbol{x}$ the matrix with the inputs and $\boldsymbol{y}$ the matrix with the outputs) where the columns of the matrix represent the variables and the rows represent the observations as following:

$\boldsymbol{x}_{[n \times p]}$, $\boldsymbol{y}_{[n \times q]}$

Where $\boldsymbol{n}$ is the number of observations, $\boldsymbol{p}$ the number of input and $\boldsymbol{q}$ the number of output variables.

The environmental variable matrix has the same form as the inputs and outputs matrixes described before:

$\boldsymbol{z}_{[n \times r]}$

Algorithm_1_simplified

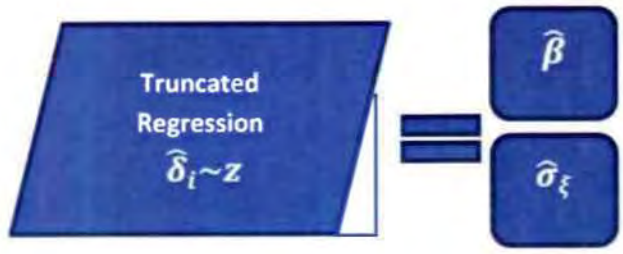
[1] The first step is about to estimate the inefficiencies $\hat{\delta}_i$ for each firm i. In step [1], DEA is used to calculate the inefficiencies and in practice the algorithm sets x and y in the DEA function (Algorithm_1_simplified, 25) and saves the estimated inefficiencies $\widehat{\delta}_i$ in d_hat (Algorithm_1_simplified, 26).

These inefficiency scores are attached to all observations (firms) with the full efficient firms taking the value of $\mathbf{1}$ and the inefficient ones values $> \mathbf{1}$ (if the DEA model is output-oriented, in the algorithm this is defined as ORIENTATION= "out"). If the DEA model is input-oriented (ORIENTATION= "in"), the full efficient firms have efficiency score equal to $\mathbf{1}$ while the inefficient ones have a score $< \mathbf{1}$.

The other choice that the user has to make is to choose between Constant Returns to Scale (CRS) and Variable Returns to Scale (VRS) which are set in the DEA function (Algorithm_1_simplified, 25) as RTS="CRS" and RTS="VRS".

[2] The second step requires the estimation of $\bar{\beta}$ and $\hat{\sigma}_{\xi}$. where $\bar{\sigma}_{\xi}$ is the estimated standard deviation of the residuals, $\bar{\beta}$ includes the estimated constant term and the effects of the environmental variables z on the inefficiency scores $\widehat{\delta}_i$ and ξ denotes the residuals.

The algorithm uses the maximum likelihood method to apply the truncated regression of z on $\bar{\delta}_i$ (the product of step [1]) resulting in the estimates $\bar{\beta}$ and the residual standard deviation estimates $\bar{\sigma}_{\varepsilon}$ (Algorithm_1_simplified, 33-41). (Algorithm_1_simplified, 33-36) is used in the output-oriented case and (Algorithm_1_simplified, 38-41) in the input-oriented case. Then, it saves the constant term and slopes at $beta\_hat$ and the standard deviation at sd_hat (Algorithm_1_simplified, 46-57).



[3.1] In this step the error term u_i is created using random draws from $N(0, \hat{\sigma}_\xi^2)$, where this distribution is left-truncated at $c = (1 - z_i\hat{\beta})$. This means that each u_i $(i = 1, \dots, m)$ is randomly picked from the distribution mentioned before but it's not allowed to pick values that fall at the left side of c on the distribution. So, for each element of u_i the distribution is "truncated" at a different point:

$$u_1 \rightarrow c = 1 - (\hat{\beta}_1 z_{1,1} + \hat{\beta}_2 z_{1\,2} + \cdots + \hat{\beta}_r z_{1\,r})$$

$$u_2 \rightarrow c = 1 - (\hat{\beta}_1 z_{2,1} + \hat{\beta}_2 z_{2\,2} + \cdots + \hat{\beta}_r z_{2,r})$$

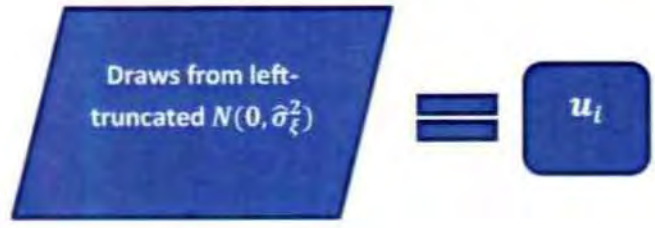
$$\vdots$$

$$u_m \rightarrow c = 1 - (\hat{\beta}_1 z_{m,1} + \hat{\beta}_2 z_{m,2} + \cdots + \hat{\beta}_r z_{m,r})$$

An easy way to draw each of the elements of u_i from a left-truncated normal distribution is using a modified transformation method (Simar and Wilson, 2007):

Let v denote a uniform distribution on $(0, 1)$, $c' = c/\hat{\sigma}_\xi$ and set $v' = \Phi(c') + [1 - \Phi(c')]v$. Now one can compute u_i as $u_i = \hat{\sigma}_\xi \Phi^{-1}(v')$.

The algorithm firstly computes the point of truncation for each observation c (Algorithm_1_simplified, 59-65) and then the object c' as described above (Algorithm_1_simplified, 67). The algorithmic notation of c and c' is c and cc respectively. These two objects are constructed outside the bootstrap loop [3] because their values don't change between different draws. Then, v and v' are computed (Algorithm_1_simplified, 70-72) inside the bootstrap loop [3] as the random number generator gives different values to the uniform distribution in every iteration. Their algorithmic notation is v and vv respectively. Finally, the error term u_i is constructed and saved in variable u (Algorithm_1_simplified, 73).



[3.2] The goal in this step is to compute the new inefficiencies δ_i^*. Each one of these is constructed as $\delta_i^* = \sum_{j=1}^{r} z_{i,j}\hat{\beta}_j + u_i$, where $i = 1, \dots, m, j = 1, \dots, r$ and m is the number of $\delta_i > 1$. Thus, for the construction of δ_i^*, one needs to choose from each environmental variable these observations that correspond to a $\delta_i > 1$. In particular each δ_i^* is computed as:

$$\delta_1^* = z_{1,1}\hat{\beta}_1 + z_{1,2}\hat{\beta}_2 + \cdots + z_{1,r}\hat{\beta}_r + u_1$$

$$\delta_2^* = z_{2,1}\hat{\beta}_1 + z_{2,2}\hat{\beta}_2 + \cdots + z_{2,r}\hat{\beta}_r + u_2$$

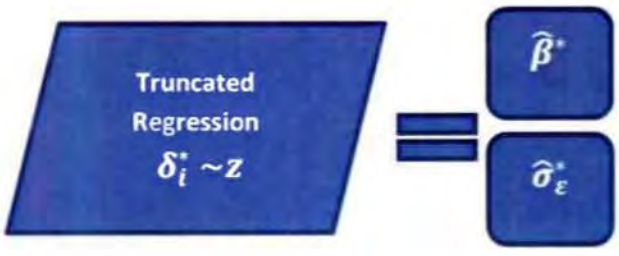
$$\vdots$$

$$\delta_m^* = z_{m,1}\hat{\beta}_1 + z_{m,2}\hat{\beta}_2 + \cdots + z_{m,r}\hat{\beta}_r + u_m$$

For this step, the algorithm uses the observation of z where $\delta_i > 1$ that was previously computed and saved in z_m_dt (Algorithm_1_simplified, 57) and multiplies them with the relevant $\hat{\beta}$ that are saved in zb (Algorithm_1_simplified, 64). Then, it adds these quantities plus the error term that computed before for this particular observation (Algorithm_1_simplified, 76) and saved the computed δ_i^* in d_star. Notice that for the construction of zb in (Algorithm_1_simplified, 64) the first element of $beta\_hat$ (the constant term) is not used.

[3.3] Using the truncated regression of z on δ_i^*, the maximum likelihood estimates $(\hat{\beta}^*, \hat{\sigma}_\varepsilon^*)$ are estimated.

The algorithm applies the truncated regression of z on δ_i^* (Algorithm_1_simplified, 79-87) using (Algorithm_1_simplified, 79-82) in the output-oriented and (Algorithm_1_simplified, 84-87) in the input-oriented case and $\hat{\beta}^*$ and $\hat{\sigma}_\varepsilon^*$ are saved as $beta\_hat\_star$ and sd_hat_star respectively (Algorithm_1_simplified, 92-93).



The bootstrap loop [3] ends forming a matrix $\boldsymbol{A}$ containing all the bootstrapped estimates $(\tilde{\boldsymbol{\beta}}^{*}, \tilde{\sigma}_{\varepsilon}^{*})$ found in every iteration of [3] at step [3.3].

Thus, at the end of every iteration after [3.3], the algorithm stores the estimated values of $\tilde{\boldsymbol{\beta}}^{*}$ and $\tilde{\sigma}_{\varepsilon}^{*}$ at $bootstrap\_beta$ and $bootstrap\_sd$ respectively (Algorithm_1_simplified, 95-98) and when the bootstrap loop [3] ends, it binds these bootstrap groups forming a big matrix of bootstrapped sets $\boldsymbol{A} = \{(\tilde{\boldsymbol{\beta}}^{*}, \tilde{\sigma}_{\varepsilon}^{*})_{i,}\}_{b=1}^{L}$ (Algorithm_1_simplified, 100).

[4] In the last step, confidence intervals are constructed for the constant term $(\boldsymbol{\beta_0})$, the coefficients $(\boldsymbol{\beta_j})$ of all environmental variables $\boldsymbol{z}$ and the standard deviation $\boldsymbol{\sigma_\varepsilon}$.

The algorithm provides two possible methods for the confidence intervals construction as well as the option to estimate both and compare the results. One has to run the algorithm choosing $ci = "basic"$ for the basic method of confidence intervals, $ci = "percentile"$ for the percentile method and $ci = "all"$ in order to compute both and compare. In case of no choice, the percentile method is set as default. Also, the significance level $alpha$ has to be chosen and again in case of no choice, $alpha = 0.05$ is set as default.

The algorithm firstly stores the appropriate lower and upper quantiles for each of the bootstrap distributions existing in $\boldsymbol{A}$ in $quantiles$(Algorithm_1_simplified, 105-109). Then, are estimated the confidence intervals (Algorithm_1_simplified, 119-160).

The basic method (Algorithm_1_simplified, 115-134) indicates the construction of the lower and the upper confidence intervals as 2 times the initial statistic of interest (for example $\boldsymbol{\beta}$) minus the upper or the lower quantile, respectively:

Lower confidence intervals for $\boldsymbol{\beta}$: $\boldsymbol{2\tilde{\beta} - q_{1-\alpha/2}^{(\beta)}}$

Upper confidence intervals for $\boldsymbol{\beta}$: $\boldsymbol{2\tilde{\beta} - q_{\alpha/2}^{(\beta)}}$

Lower confidence intervals for $\boldsymbol{\sigma_\varepsilon}$: $\boldsymbol{2\tilde{\sigma}_\varepsilon - q_{1-\alpha/2}^{(\sigma_\varepsilon)}}$

Upper confidence intervals for $\boldsymbol{\sigma_\varepsilon}$: $\boldsymbol{2\tilde{\sigma}_\varepsilon - q_{\alpha/2}^{(\sigma_\varepsilon)}}$

Where $\boldsymbol{q_{\alpha/2}}$ and $\boldsymbol{q_{1-\alpha/2}}$ are the lower and the upper quantiles from the sorted bootstrap distribution of the statistic of interest. The algorithm calculates and stores them in the object b_ci (Algorithm_1_simplified, 116-121). Finally, the results are printed out on the screen (Algorithm_1_simplified, 124-133).

The same procedure is followed for the percentile confidence intervals (Algorithm_1_simplified, 137-156). The only difference here is the calculation method that indicates that the lower and upper bounds are simply the lower and upper quantiles of the sorted bootstrap distribution of the statistic of interest respectively:

Lower confidence intervals for $\boldsymbol{\beta}$: $q_{\alpha/2}^{(\beta)}$

Upper confidence intervals for $\boldsymbol{\beta}$: $q_{1-\alpha/2}^{(\beta)}$

Lower confidence intervals for $\boldsymbol{\sigma_\varepsilon}$: $q_{\alpha/2}^{(\sigma_\varepsilon)}$

Upper confidence intervals for $\boldsymbol{\sigma_\varepsilon}$: $q_{1-\alpha/2}^{(\sigma_\varepsilon)}$

The algorithm calculates and then stores them in the object p_ci (Algorithm_1_simplified, 142-147). Finally, the results are printed out on the screen (Algorithm_1_simplified, 146-155).

Algorithm_2_simplified

[1] The first step is about to estimate the inefficiencies $\boldsymbol{\delta_i}$ for each firm $\boldsymbol{i}$. In step [1], DEA is used to calculate the inefficiencies and in practice the algorithm sets $\boldsymbol{x}$ and $\boldsymbol{y}$ in the DEA function (Algorithm_2_simplified, 25) and saves the estimated inefficiencies $\widehat{\boldsymbol{\delta}}_i$ in d_hat (Algorithm_2_simplified, 26).

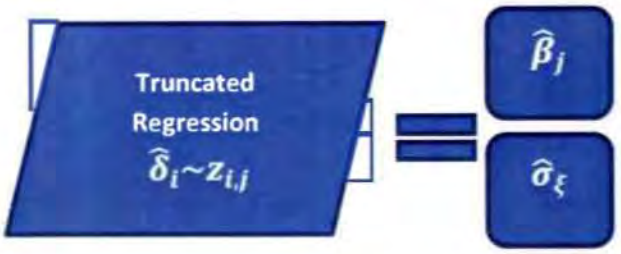
These inefficiency scores are attached to all observations (firms) with the full efficient firms taking the value of $\mathbf{1}$ and the inefficient ones values $> \mathbf{1}$ (if the DEA model is output-oriented, in the algorithm this is defined as ORIENTATION= "out"). If the DEA model is input-oriented (ORIENTATION= "in"), the full efficient firms have efficiency score equal to $\mathbf{1}$ while the inefficient ones have a score $< \mathbf{1}$.

The other choice that the user has to make is to choose between Constant Returns to Scale (CRS) and Variable Returns to Scale (VRS) which are set in the DEA function (Algorithm_2_simplified, 25) as RTS="CRS" and RTS="VRS".

[2] The second step requires the estimation of $\tilde{\boldsymbol{\beta}}$ and $\hat{\sigma}_\xi$, where $\hat{\sigma}_\xi$ is the estimated standard deviation of the residuals, $\tilde{\boldsymbol{\beta}}$ includes the estimated constant term and the effects of the environmental variables $\boldsymbol{z}$ on the inefficiency scores $\tilde{\boldsymbol{\delta}}_i$ and ξ denotes the residuals.

The algorithm uses the maximum likelihood method to apply the truncated regression of $\boldsymbol{z}$ on $\tilde{\boldsymbol{\delta}}_i$ (the product of step [1]) resulting in the estimates $\tilde{\boldsymbol{\beta}}$ and the residual standard

deviation estimates $\tilde{\sigma}_{\varepsilon}$ (Algorithm_2_simplified, 33-47). (Algorithm_2_simplified, 33-36) is used in the output-oriented case and (Algorithm_2_simplified, 38-41) in the input-oriented case. Then, it saves the constant term and slopes at *beta_hat* and the standard deviation at *sd_hat* (Algorithm_2_simplified, 46-47).



[3.1] In this step the error term u_i is created using random draws from $N(0, \tilde{\sigma}_{\xi}^2)$, where this distribution is left-truncated at $c = (1 - z_{i,j}\hat{\beta}_j)$. This means that each u_i $(i = 1, \dots, n)$ is randomly picked from the distribution mentioned before but it's not allowed to pick values that fall at the left side of c on the distribution. So, for each element of u_i the distribution is "truncated" at a different point:

$$u_1 \rightarrow c = 1 - (\hat{\beta}_1 z_{1,1} + \hat{\beta}_2 z_{1\,2} + \cdots + \hat{\beta}_r z_{1\,r})$$

$$u_2 \rightarrow c = 1 - (\hat{\beta}_1 z_{2,1} + \hat{\beta}_2 z_{2,2} + \cdots + \hat{\beta}_r z_{2,r})$$

$$\vdots$$

$$u_n \rightarrow c = 1 - (\hat{\beta}_1 z_{n\,1} + \hat{\beta}_2 z_{n\,2} + \cdots + \hat{\beta}_r z_{n,r})$$

An easy way to draw each of the elements of u_i from a left-truncated normal distribution is using a modified transformation method (Simar and Wilson, 2007):

Let v denote a uniform distribution on $(0, 1)$, $c' = c/\tilde{\sigma}_{\xi}$ and set $v' = \Phi(c') + [1 - \Phi(c')]v$. Now one can compute u_i as $u_i = \tilde{\sigma}_{\xi}\Phi^{-1}(v')$.

The algorithm firstly computes the point of truncation for each observation c (Algorithm_2_simplified, 56-62) and then the object c' as described above (Algorithm_2_simplified, 64). The algorithmic notation of c and c' is c and cc respectively. These two objects are constructed outside the bootstrap loop [3] because their values don't change between different draws. Then, v and v' are computed (Algorithm_2_simplified, 68-69) inside the bootstrap loop [3] as the random number generator gives different values to the uniform distribution in every iteration. Their algorithmic notation is v and vv respectively. Finally, the error term u_i is constructed and saved in variable u (Algorithm_2_simplified, 70).

[3.2] The goal in this step is to compute the new inefficiencies δ_i^*. Each one of these is constructed as $\delta_i^* = \sum_{j=1}^{r} z_{i,j}\bar{\beta}_j + u_i$, where $i = 1, \dots, n, j = 1, \dots, r$. In particular each δ_i^* is computed as:

$$\delta_1^* = z_{1,1}\beta_1 + z_{1,2}\beta_2 + \cdots + z_{1,r}\beta_r + u_1$$

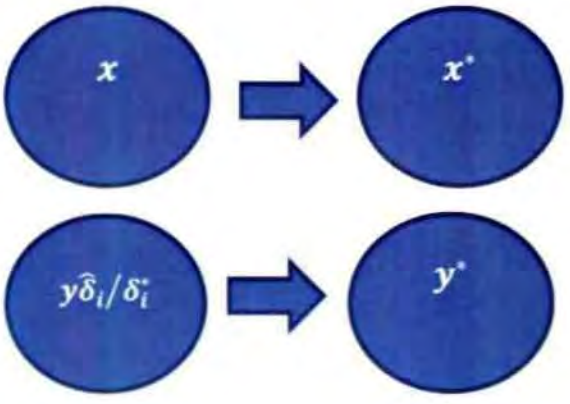
$$\delta_2^* = z_{2,1}\bar{\beta}_1 + z_{2,2}\bar{\beta}_2 + \cdots + z_{2,r}\bar{\beta}_r + u_2$$

$$\vdots$$

$$\delta_n^* = z_{n,1}\beta_1 + z_{n,2}\beta_2 + \cdots + z_{n,r}\bar{\beta}_r + u_n$$

For this step, the algorithm multiplies each observation of z with the relevant $\bar{\beta}$ and saves them in zb (Algorithm_2_simplified, 56-62). Then, it adds these quantities plus the error term that computed before for this particular observation (Algorithm_2_simplified, 73) and saved the computed δ_i^* in d_star. Notice that for the construction of zb in (Algorithm_2_simplified, 56-62) the first element of $beta\_hat$ (the constant term) is not used.

[3.3] Sets the variable x into a new object named x_star and construct the object y_star multiplying every observation of y with those of $\bar{\delta}_i$ and divide each of these quantities with the corresponding observation of δ_i^*.



[3.4] This step estimates the inefficiencies δ_i^* for each firm i. DEA is used to calculate the inefficiencies and in practice the algorithm sets x^* and y^* in the DEA function (Algorithm_2_simplified, 80) and saves the estimated inefficiencies δ_i^* in d_hat_star (Algorithm_2_simplified, 81).

These inefficiency scores are attached to all observations (firms) with the full efficient firms taking the value of $\mathbf{1}$ and the inefficient ones values $> \mathbf{1}$ (if the DEA model is output-oriented, in the algorithm this is defined as ORIENTATION= "out"). If the DEA model is input-oriented (ORIENTATION= "in"), the full efficient firms have efficiency score equal to $\mathbf{1}$ while the inefficient ones have a score $< \mathbf{1}$.

The other choice that the user has to make is to choose between Constant Returns to Scale (CRS) and Variable Returns to Scale (VRS) which are set in the DEA function (Algorithm_2_simplified, 80) as RTS="CRS" and RTS="VRS".

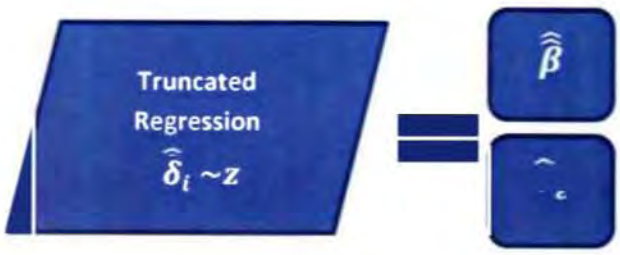
[4] Here the bias-corrected estimator $\hat{\boldsymbol{\delta}}_i$ is computed subtracting the bias from the initial estimator $\boldsymbol{\delta}_i$, where the bias of every observation of $\boldsymbol{\delta}_i$ is estimated subtracting $\boldsymbol{\delta}_i$ from the mean of the bootstrap estimators $\boldsymbol{\delta}_i^*$.

The algorithm first subtracts each observation of d_hat from those of $mean\_di\_hat\_star$ and stores them in the object $bias$ (Algorithm_2_simplified, 94-95), where in the object $mean\_di\_hat\_star$ are stored the mean values of all the bootstrap loops for every observation of $\bar{\boldsymbol{\delta}}$ (Algorithm_2_simplified, 90-92). Then it subtracts the bias stored in $bias$ from d_hat and saves the resulted bias-corrected estimator $\hat{\boldsymbol{\delta}}_i$ in d_2hat.

[5] In this step the estimators $\hat{\hat{\boldsymbol{\beta}}}$ and $\hat{\hat{\sigma}}_{\varepsilon}$ are computed, where $\hat{\hat{\sigma}}_{\varepsilon}$ is the estimated standard deviation of the residuals, $\hat{\hat{\boldsymbol{\beta}}}$ includes the estimated constant term and the effects of the environmental variables $\boldsymbol{z}$ on the inefficiency scores $\hat{\hat{\boldsymbol{\delta}}}_i$ and ε denotes the residuals.

The algorithm uses the maximum likelihood method to apply the truncated regression of $\boldsymbol{z}$ on $\hat{\delta}_i$ (the product of step [4]) resulting in the estimates $\hat{\hat{\boldsymbol{\beta}}}$ and the residual standard deviation estimates $\hat{\hat{\sigma}}_{\varepsilon}$ (Algorithm_2_simplified, 100-114). (Algorithm_2_simplified, 100-

103) is used in the output-oriented case and (Algorithm_2_simplified, 105-108) in the input-oriented case. Then, it saves the constant term and slopes at $beta\_2hat$ and the standard deviation at sd_2hat (Algorithm_2_simplified, 113-114).



[6.1] In this step the error term u_i is created using random draws from $N(0, \hat{\hat{\sigma}}_\varepsilon^2)$, where this distribution is left-truncated at $c_i = (1 - z_{ij}\hat{\beta}_j)$. This means that each u_i $(i = 1, \dots, n)$ is randomly picked from the distribution mentioned before but it's not allowed to pick values that fall at the left side of c on the distribution. So, for each element of u_i the distribution is "truncated" at a different point:

$$u_1 \rightarrow c = 1 - \left(\hat{\hat{\beta}}_1 z_{1,1} + \hat{\hat{\beta}}_2 z_{1\,2} + \cdots + \hat{\hat{\beta}}_r z_{1,r}\right)$$

$$u_2 \rightarrow c = 1 - \left(\hat{\hat{\beta}}_1 z_{2,1} + \hat{\hat{\beta}}_2 z_{2,2} + \cdots + \hat{\hat{\beta}}_r z_{2,r}\right)$$

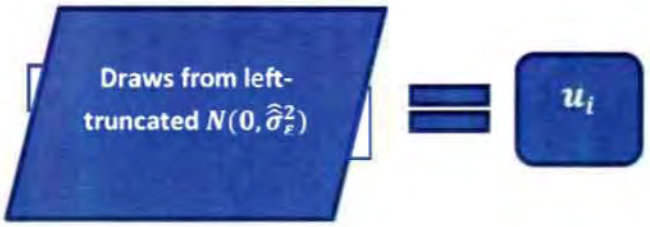
$$\vdots$$

$$u_n \rightarrow c = 1 - \left(\hat{\hat{\beta}}_1 z_{n,1} + \hat{\hat{\beta}}_2 z_{n,2} + \cdots + \hat{\hat{\beta}}_r z_{n,r}\right)$$

An easy way to draw each of the elements of u_i from a left-truncated normal distribution is using a modified transformation method (Simar and Wilson, 2007):

Let v denote a uniform distribution on $(0, 1)$, $c' = c/\hat{\hat{\sigma}}_\varepsilon$ and set $v' = \Phi(c') + [1 - \Phi(c')]v$. Now one can compute u_i as $u_i = \hat{\hat{\sigma}}_\varepsilon \Phi^{-1}(v')$.

The algorithm firstly computes the point of truncation for each observation c (Algorithm_2_simplified, 124-130) and then the object c' as described above (Algorithm_2_simplified, 132). The algorithmic notation of c and c' is c_2 and cc_2 respectively. These two objects are constructed outside the bootstrap loop [6] because their values don't change between different draws. Then, v and v' are computed (Algorithm_2_simplified, 136-137) inside the bootstrap loop [6] as the random number generator gives different values to the uniform distribution in every iteration. Their algorithmic notation is $v\ 2$ and vv_2 respectively. Finally, the error term u_i is constructed and saved in the object u_2 (Algorithm_2_simplified, 138).



[6.2] The goal in this step is to compute the new inefficiencies δ_i^{**}. Each one of these is constructed as $\delta_i^{**} = \sum_{j=1}^{r} z_{i,j}\hat{\beta}_j + u_i$, where $i = 1, \dots, n, j = 1, \dots, r$. In particular each δ_i^{**} is computed as:

$$\delta_1^{**} = z_{1\,1}\hat{\hat{\beta}}_1 + z_{1\,2}\hat{\hat{\beta}}_2 + \cdots + z_{1\,r}\hat{\hat{\beta}}_r + u_1$$

$$\delta_2^{**} = z_{2,1}\hat{\hat{\beta}}_1 + z_{2,2}\hat{\hat{\beta}}_2 + \cdots + z_{2,r}\hat{\hat{\beta}}_r + u_2$$

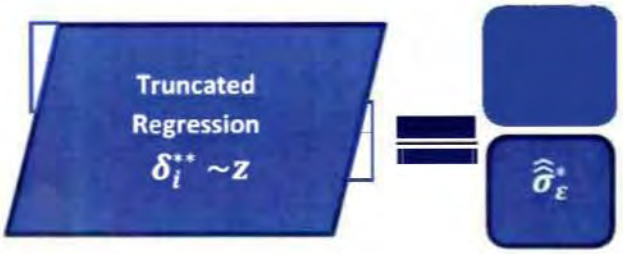
$$\vdots$$

$$\delta_n^{**} = z_{n\,1}\hat{\hat{\beta}}_1 + z_{n\,2}\hat{\hat{\beta}}_2 + \cdots + z_{n,r}\hat{\hat{\beta}}_r + u_n$$

For this step, the algorithm constructs the object zb_2 as the sum of the multiplication between $\hat{\hat{\beta}}$ and z (Algorithm_2_simplified, 124-130) and adds the error term that computed before for each particular observation (Algorithm_2_simplified, 141) and saved the computed δ_i^{**} in d_2star. Notice that for the construction of zb_2 in (Algorithm_2_simplified, 124-130) the first element of $beta\_2hat$ (the constant term) is not used.

[6.3] Using the truncated regression of z on δ_i^{**}, the maximum likelihood estimates $(\hat{\hat{\beta}}^{*}, \hat{\hat{\sigma}}_{\varepsilon}^{*})$ are estimated.

The algorithm applies the truncated regression of z on δ_i^{**} (Algorithm_2_simplified, 144-152) using (Algorithm_2_simplified, 144-147) in the output-oriented and (Algorithm_2_simplified, 149-152) in the input-oriented case and $\hat{\hat{\beta}}^{*}$ and $\hat{\hat{\sigma}}_{\varepsilon}^{*}$ are saved as $beta\_2hat\_star$ and sd_2hat_star respectively (Algorithm_2_simplified, 157-158).



The bootstrap loop [6] ends forming a matrix $\boldsymbol{C}$, containing all the bootstrapped estimates $\left(\widehat{\boldsymbol{\beta}}^{*}, \widehat{\sigma}_{\varepsilon}^{*}\right)$ found in every iteration of [6] at step [6.3].

Thus, at the end of every iteration after [6.3], the algorithm stores the estimated values of $\widehat{\boldsymbol{\beta}}^{*}$and $\widehat{\sigma}_{\varepsilon}^{*}$ at $bootstrap\_beta$ and $bootstrap\_sd$ respectively (Algorithm_2_simplified, 160-163) and when the bootstrap loop [6] ends, it binds these bootstrap groups forming a big matrix of bootstrapped sets $\boldsymbol{C} = \{\left(\widehat{\boldsymbol{\beta}}^{*}, \widehat{\sigma}_{\varepsilon}^{*}\right)_{b}\}_{b=1}^{b_{2}}$ (Algorithm_2_simplified, 165).

[7] In the last step, confidence intervals are constructed for the constant term $(\boldsymbol{\beta}_0)$, the coefficients $(\boldsymbol{\beta}_j)$ of all environmental variables $\boldsymbol{z}_j$ and the standard deviation $\boldsymbol{\sigma}_{\boldsymbol{\varepsilon}}$.

The algorithm provides two possible methods for the confidence intervals construction as well as the option to estimate both and compare the results. One has to run the algorithm choosing $ci = "basic"$ for the basic method of confidence intervals, $ci = "percentile"$ for the percentile method and $ci = "all"$ in order to compute both and compare. In case of no choice, the percentile method is set as default. Also, the significance level $alpha$ has to be chosen and again in case of no choice, $alpha = 0.05$ is set as default.

The algorithm firstly stores the appropriate lower and upper quantiles for each of the bootstrap distributions existing in $\boldsymbol{C}$ in $quantiles$(Algorithm_2_simplified, 170-174). Then, the confidence intervals are estimated (Algorithm_2_simplified, 180-217).

The basic method (Algorithm_2_simplified, 180-197) indicates the construction of the lower and the upper confidence intervals as 2 times the initial statistic of interest (for example $\boldsymbol{\beta}$) minus the upper or the lower quantile, respectively:

Lower confidence intervals for $\boldsymbol{\beta}$: $\boldsymbol{2\widehat{\beta} - q_{1-\alpha/2}^{(\beta)}}$

Upper confidence intervals for $\boldsymbol{\beta}$: $\boldsymbol{2\widehat{\beta} - q_{\alpha/2}^{(\beta)}}$

Lower confidence intervals for $\boldsymbol{\sigma_{\varepsilon}}$: $\boldsymbol{2\widehat{\sigma}_{\varepsilon} - q_{1-\alpha/2}^{(\sigma_{\varepsilon})}}$

Upper confidence intervals for $\boldsymbol{\sigma_{\varepsilon}}$: $\boldsymbol{2\widehat{\sigma}_{\varepsilon} - q_{\alpha/2}^{(\sigma_{\varepsilon})}}$

Where $\boldsymbol{q_{\alpha/2}}$ and $\boldsymbol{q_{1-\alpha/2}}$ are the lower and the upper quantiles from the sorted bootstrap distribution of the statistic of interest. The algorithm calculates and stores them in the object b_ci (Algorithm_2_simplified, 181-186). Finally, the results are printed out on the screen (Algorithm_2_simplified, 192).

The same procedure is followed for the percentile confidence intervals (Algorithm_2_simplified, 200-217). The only difference here is the calculation method that indicates that the lower and upper bounds are simply the lower and upper quantiles of the sorted bootstrap distribution of the statistic of interest respectively:

Lower confidence intervals for $\boldsymbol{\beta}$: $q_{\alpha/2}^{(\beta)}$

Upper confidence intervals for $\boldsymbol{\beta}$: $q_{1-\alpha/2}^{(\beta)}$

Lower confidence intervals for σ_{ε}: $q_{\alpha/2}^{(\sigma_{\varepsilon})}$

Upper confidence intervals for σ_{ε}: $q_{1-\alpha/2}^{(\sigma_{\varepsilon})}$

The algorithm calculates and then stores them in the object p_ci (Algorithm_2_simplified, 201-206). Finally, the results are printed out on the screen (Algorithm_2_simplified, 216).

3.5. Application

In this empirical application, the algorithms described before are used to examine the effects of Advertising Expense and Communications Expense on the performance of banks operating in U.S. More specifically, the two-stage semi-parametric approach is used in the first-stage to estimate the inefficiency of the U.S. bank sector during the years 2006-2013, based on the sample data of 148 banks for this period. In the second stage, the effects of Advertising Expense and Communications Expense on these estimated inefficiencies are calculated constructing confidence intervals for each one of them, as well as some other interesting statistics to make the analysis more robust.

The technical efficiency that is estimated in the first-stage using DEA, uses the Total Assets, Total Deposits and the number of Employees as inputs in the production function of the banks and the Total Securities and Gross Loans as the outputs of the banks' performance. The covariates (environmental variables) that mentioned before are assumed to affect banks' performance (measured by technical efficiency) but not the Production Possibility Function (PPF) described by the frontier line. This is because of the assumptions where the algorithms are based on, about the DGP. This way, the environmental variables don't take part in the DEA stage where the frontier is estimated, but only in the second-stage of the analysis. This is supported by the assumption (A1).

At this point, a small discussion about the DGP, interpreting each one of its assumptions would be helpful for better understanding.

Reading (1) from right to left, one can see that the environmental variables $\boldsymbol{z}$ are independent. So, the decision of a bank about the Advertising and Communications spending is clearly under the bank's control. On the other hand, the technical efficiency depends on $\boldsymbol{z}$, and this is the reason for the second-stage analysis. This means that a bank can change its performance (increase or decrease its efficiency) alternating the money it spends for Advertising and Communications but without overcoming its capability (PPF). For the next function, is important to remind that the outputs $\boldsymbol{y}$ in terms of polar coordinates is described as $(\boldsymbol{\eta_i}, \boldsymbol{\delta_i})$ where $\boldsymbol{\eta_i}$ is the angle and $\boldsymbol{\delta_i}$ is the distance that determines the efficiency Thus, this function indicates that the inputs x depend on the efficiency and so on the environmental variables $\boldsymbol{z}$ and also, $\boldsymbol{\eta_i}$ (let's say, a part of $\boldsymbol{y}$) depends on the efficiency

and so on $\boldsymbol{z}$. Interpreting this one can say that a bank makes decisions about its inputs (Assets, Deposits, Employees) based on the money they have spent for Advertising and Communications and so based on the efficiency that these spending had. The situation about the outputs $\boldsymbol{y}$ is not exactly like this, as a "part" of this (in polar coordinates, the angle $\boldsymbol{\eta_i}$) depends also on the same $\boldsymbol{z}$ and $\boldsymbol{\delta_i}$ as the inputs but not the whole $\boldsymbol{y}$. This means that the bank can partly decide about its outputs (Securities and Loans) but a part of these is "automatically" created as the output of the spending mentioned before, this part of the output is the efficiency $\boldsymbol{\delta_i}$. The whole $\boldsymbol{y}$ together with $\boldsymbol{x}$ are these that determine the capability of the bank (PPF), so given certain inputs and outputs the bank sets itself a maximum of achievable efficiency.

A short interpretation of the assumptions (A1-A8) is considered also useful.

What assumptions A1 and A2 say, is that while the inputs and outputs (Assets, Deposits, Employees, Securities and Loans) are these that determine the maximum achievable efficiency, the environmental variables (Advertising and Communications expense) are those that determine its level within the boundaries that $\boldsymbol{x}$ and $\boldsymbol{y}$ set. The assumption A3 determines the kind of the error term in the second-stage regression.

The assumptions A4-A6 indicate that the inputs and outputs of a bank are depended to each other meaning that P also depends on the various feasible combinations of $\boldsymbol{x}$ and $\boldsymbol{y}$. There are no "free lunches", so for example a bank cannot give any Loans while its Deposits are zero. The last assumption is about the "free disposability", that says a banks can give, for example, a Loan of 1000 $ even if its Deposits are more (say 2000$). The last two assumptions A7 and A8 ensure the consistency of $\boldsymbol{P}$ and $\boldsymbol{\delta_i}$.

For the empirical application, the two algorithms (Algorithm_1 and Algorithm_2) were implemented give the three inputs and the two outputs that mentioned before and various combinations of the covariates in order to examine all the possible models. In addition, both the basic and the percentile confidence intervals are constructed setting ci equal to "all", not only for the coefficients of the covariates and the standard deviation but also for a few other statistics (sigma, P-value, R^2, AIC, BIC and RMSE) for a more robust analysis.

All the models consider Constant Returns to Scale (CRS) as the classic microeconomic theory indicates for the second stage. The DEA model in the first-stage is output oriented in all cases as the application examines the technical efficiency. An extra parameter called seed is set equal to 1 for all the models for easy re-productivity, except some models where numerical problems occur. In these cases, the seed changed adding 1 each time until there are no numerical problems in the calculations for any loop.

The number of iterations for every bootstrap loop ($\boldsymbol{L}$ and $\boldsymbol{L_2}$) for the construction of the confidence intervals is set equal to 2000 iterations as Simar and Wilson (2007) do. According to Hall (1986), 1000 is a sufficient number of replication for the confidence-intervals construction, but this number is doubled here for more accurate results. A sufficient number

of replications for the estimations of the bias-corrected efficiencies are 100, as Simar and Wilson (2007) suggest, but here 500 iterations (L_1) are used for better results.

All the tables with all the possible models are presented in the Appendix. For our convenience, lets name the models by the numbers in the name of the external variable in each model. For example, the model with ln(Advertising.Expense), ln(Advertising.Expense)^2 and ln(Communications.Expense) as environmental variables is the model 21. The first number represents the maximum power of the variable ln(Advertising.Expense) and the second number the maximum power of the variable ln(Communications.Expense).

Here is presented a Table with the summary of all the tables (except the model where the covariates are in levels rather than logarithms) with reference to the signs and the statistical significance:

Table 4

Algorithm_1	Signs (percentile)	Signs
C	Insignificant	- 13/15
ln(Advertising.Expense)	+ 8/8	+ 1/1
ln(Advertising.Expense)^2	+ 4/4	+ 1/1
ln(Advertising.Expense)^3	Insignificant	Insignificant
ln(Communications.Expense)	+ 6/6	+ 2/2
ln(Communications.Expense)^2	- 2/3	Insignificant
ln(Communications.Expense)^3	- 1/1	Insignificant

Table 5

Algorithm_2	Signs
C	+ 13/15[6]
ln(Advertising.Expense)	+ 12/12
ln(Advertising.Expense)^2	+ 7/7
ln(Advertising.Expense)^3	- 2/2
ln(Communications.Expense)	+ 11/11
ln(Communications.Expense)^2	- 5/6
ln(Communications.Expense)^3	- 1/1

From the original tables in the Appendix or the summary table above, one can easily see that the results from Algorithm_1 are much worse than those from Algorithm_2 as expected. Especially the confidence intervals constructed with the basic method are very far from reality as three of the six variables found to be insignificant in every model, the rest of the variables found also insignificant (with one or two exceptions) and the constant in every model falls outside of the confidence interval.

[6] This ratio is only about the basic confidence intervals for the constant but the majority of the positive signs in this case in not very meaningful as in all of the models the mean is outside of the basic confidence interval bounds. In the case of the percentile confidence intervals, all the intervals contain zero, so they was found to be insignificant. The same happens with Algorithm_1.

The results of Algorithm_1 as computed with the percentile confidence intervals are somehow better that those from the basic method, they generally agree with the results of Algorithm_2 but with less models to confirm the positive or negative effects on the inefficiencies and the constant term along with the log of Advertising Expense in the third power was found insignificant in every model.

All these imperfections on the results of Algorithm_1 were expected as Algorithm_1 regress the covariates on the uncorrected inefficiency scores in contrast with Algorithm_2 where the covariates are regressed on the bias-corrected inefficiencies $\hat{\hat{\delta}}_i$. Also, the bad results from the basic method were probably caused because of the more complicated way of estimation rather than simply taking the quantiles of the bootstrap distribution as the percentile method does. In the basic method, extreme results may be produced in cases where the initial estimate happened to be an extreme value (whether too high or too low).

The results from Algorithm_2 clearly show that there are positive effects of the log of Advertising Expense, the log of Advertising Expense in the second power and the log of Communications Expense on the inefficiencies, negative effects of the log of Communications Expense in the second power and probably negative effect of the log of Advertising Expense and Communications Expense in the third power. The constant term is not very meaningful as it either falls outside of the confidence interval bounds or it is insignificant.

The conclusions are based on Algorithm_2 rather than Algorithm_1 as its results are much more precise. This is clear not only form this empirical study but also as the Monte Carlo simulation and the empirical example of these models shown in the original paper of Simar and Wilson (2007).

A more focused investigation of the models is essential, so the models where we have a "case" are examined explicitly. A "case" is considered a model where all the variables taking part in the model are statistically significant (only the Algorithm_2 is examined for perfect cases). A statistically significant variable is considered one that its confidence interval does not contain zero. In the tables, for every confidence interval there is the sign "!" for a significant variable and the sign "X" for an insignificant one (where the interval contains zero).

For extra support of the results, the significance is also tested looking at the P-values of the variables, where the bootstrap mean of each P-value is tested and "***" is attached when the bootstrap mean of the corresponding P-value is < 0.01, "**" when it is between 0.01 and 0.05, "*" when it is between 0.05 and 0.1 and there is nothing when the P-value is greater than 10%. Also, the lower values of sigma, AIC, BIC and RMSE indicate a better model as well as the higher value of R^2.

Based on the following four statistics (sigma, AIC, BIC and RMSE), the best four models are 22, 20, 31 and 30. Sigma, AIC and BIC, they all agree at this particular hierarchy of the models while RMSE indicates those four but with different order in the three last places, making the model 22 the best model in any case. The R^2 indicates a totally different order of

the best models but the R^2 values are quite close to each other (between 0.41 and 0.54), so the most preferable models are those that the first four criteria indicate.

These four models, they all agree that the signs of ln(Advertising.Expense) and its second power are positive, its third power is negative, ln(Communications.Expense) is positive and its second power is negative. The same signs are indicated also by the majority of the models, not only these four best. The variable ln(Communications.Expense)^3 is insignificant in all cases except the model where only Communications Expense raised in the second and third power. In this case, the pure log and the second power are positive while the third power is negative in their signs.

The final conclusion is maid based on Algorithm_2 as its results are better that those from Algorithm_1 and the confidence intervals estimated with the basic method as this is a better approach in comparison with the simples percentile method, that is also suggested in the original paper of Simar and Wilson (2007). The models that were chosen to make inference are the model 22 as this was characterized as the best model, and the model 31, one of the four best models. The model 30 is severally and jointly interpreted with the model 03 as special cases where only one of the two environmental variables participates. Notice that model 03 is not one of those models indicated by the minimization of the statistics mentioned before, but is the best model according to the R^2 criterion.

Interpreting the model 22, one can say that:

Increasing the Advertising Expense by 1%, the inefficiency of the bank increases by 0.1005. This number varies between the 95% confidence interval [0.02582, 0.1637].

Keep increasing the Advertising Expense by 1%, the inefficiency of the bank increases by 0.04346. This number varies between the 95% confidence interval [0.03038, 0.05812]

Increasing the Communications Expense by 1%, the inefficiency of the bank increases by 0.34947. This number varies between the 95% confidence interval [0.24521, 0.4601].

Keep increasing the Communications Expense by 1%, the inefficiency of the bank decreases by 0.05396. This number varies between the 95% confidence interval [0.03795, 0.07244][7].

Interpreting the model 31, one can say that:

Increasing the Advertising Expense by 1%, the inefficiency of the bank increases by 0.2007. This number varies between the 95% confidence interval [0.11518, 0.26766].

Keep increasing the Advertising Expense by 1%, the inefficiency of the bank increases by 0.06711. This number varies between the 95% confidence interval [0.04419, 0.09776]

[7] The actual coefficient in this case is -0.05396 and the confidence interval [-0.07244, -0.03795]. For better interpretation, the negative sign is omitted and is translated as a decrease. Thus, the confidence interval is inverted with positive signs.

Insisting to increase the Advertising Expense by 1%, the inefficiency of the bank decreases by 0.00674. This number varies between the 95% confidence interval [0.00404, 0.01024].

Increasing the Communications Expense by 1%, the inefficiency of the bank increases by 0.07011. This number varies between the 95% confidence interval [0.01045, 0.12406].

In the special cases where only one of the two types of expense is used as explanatory variables, the results are as follows:

Interpreting the model 30, one can say that:

Increasing the Advertising Expense by 1%, the inefficiency of the bank increases by 0.22766. This number varies between the 95% confidence interval [0.1407, 0.29301].

Keep increasing the Advertising Expense by 1%, the inefficiency of the bank increases by 0.07659. This number varies between the 95% confidence interval [0.05245, 0.10741]

Insisting to increase the Advertising Expense by 1%, the inefficiency of the bank decreases by 0.00715. This number varies between the 95% confidence interval [0.00431, 0.01075].

Interpreting the model 03, one can say that:

Increasing the Communications Expense by 1%, the inefficiency of the bank increases by 0.37014. This number varies between the 95% confidence interval [0.03767, 0.67544].

Keep increasing the Communications Expense by 1%, the inefficiency of the bank increases by 0.19014. This number varies between the 95% confidence interval [0.08583, 0.29781]

Insisting to increase the Communications Expense by 1%, the inefficiency of the bank decreases by 0.0201. This number varies between the 95% confidence interval [0.01126, 0.02963].

Comparing the model 30 with the model 31, the results are quite the same, confirming the final conclusion. In the case when the only expense is made is for Communications, the bank has to spend more money in order to reach the turning point where its inefficiency will start to decrease.

4. Results and Discussion

To sum up, in this thesis three independent R codes (max_info_clean_data, create_balanced_panel_data and create_variables) and four functions (Algorithm_1_simplified, Algorithm_2_simplified, Algorithm_1 and Algorithm_2) were constructed and given written in the Appendix as well as in the CD rom.

Using the code named max_info_clean_data, one can clean the raw data and find various samples to choose between the original vast amount of data. This code finds all the possible combinations of balanced panel samples and ranks them from the one that has the greatest amount of information to the one with the less.

Using the code named create_balanced_panel_data, one can practically create the chosen balanced panel sample and save it in his/her own computer in txt form.

The third code named create_variables, is used to create all the possible combinations of variable sets that will be used later to run the algorithms. The variable sets contain the logs of the variables Advertising Expense and Communications Expense as is but also raised in the second and the third power.

The Algorithm_1_simplified and Algorithm_2_simplified are the exact realizations of the algorithms presented by Simar and Wilson (2007), as they result only in the confidence intervals for the coefficients of the covariates and the standard deviation. They were used for educational reasons as it's much easier to read and understand these short codes rather than their much longer extensions Algorithm_1 and Algorithm_2. The last two algorithms work exactly the same way. The only difference is that they can also estimate confidence intervals for some extra statistics that the researcher may find useful in practice.

Results from the empirical economic application:

The empirical application showed that the money the banks operating in U.S. spend for advertising and communications have a significant effect on their performance where their performance was measured by their inefficiency scores

The results indicate that while a bank start spending money for advertising, its inefficiency grows (ln(Advertising.Expense) has a positive sign) increasingly (ln(Advertising.Expense)^2 has a positive sign too). Only after a sufficient amount of money for advertising their inefficiency will start to decline (ln(Advertising.Expense)^3 has a negative sign) and so their performance will increase. As for the money the banks spend for communications, the results showed that in the beginning their inefficiency increase (ln(Communications.Expense) has a positive sign) but as more money are spent for communications, they can finally start to overcome and their inefficiency decreases (ln(Communications.Expense)^2 has a negative sign). The special case when the only money spend are for communications, they have to spend even more money in communications to start decrease their inefficiency, so to perform better. This is probably because of the lack of the important contribution of advertising.

Thus based on this particular sample of banks operating in U.S. covering the years 2006-2013, one can conclude that the money the banks spend for advertising and communications are surely worthwhile, but only in the case when enough money are supposed to be spend. Otherwise, a small amount of money will increase the bank's inefficiency and will have a negative impact on their performance. Also, a good strategy is to spend money both for advertising and communications rather than only in one of them in order to be more effective.

5. Statement of Limitations

5.1. Alternatives

Alternatively, one could approach the analysis from another point, without changing the model's aim. For example, in the second-stage regression, different models specifying fixed or random effects rather than a simple pooled panel regression could have bring interesting results for examination.

Moreover, a full log transformation of the inefficiency scores too could be interesting as we could see the percentage change in the inefficiency as a result of the 1% change in the money spending.

5.2. Weaknesses

As it is common in the research, no model is perfect. The present study has also many weaknesses that should be taken into consideration for further research.

First of all, the run time of the algorithms, especially for Algorithm_1 and Algorithm_2 is very long. Depending on the number of the statistics one wants to estimate, the number of iterations of the loop, the length of the data and mainly the number of variables, especially the environmental. A faster computation of the same algorithms could be done in a faster programming language or even in R using some routines that could save time.

In addition, the final models could be analyzed further checking for specification errors, normality assumptions, and the general assumptions for the residuals. An econometric view on the final models could make the analysis more robust and reliable. Also, some kind of survival analysis and different model comparison could be also done. Some preliminary test for stationarity as well as test for the stability of the results would be a good support of the findings.

6. Conclusion

6.1. Contributions

The aim of this thesis is to review this specific field of production economics, and more precisely to analyze a specific tool commonly used in this area. The resulted algorithms of

this study can be freely used from anyone who is interested in, and help next researchers and students to apply models like this in a click of the mouse rather than spending hours of computing to do this.

The codes max_info_clean_data, create_balanced_panel_data and create_variables are not general purpose codes. They are used explicitly for the needs of this thesis, and an exact replication of these on other data rather than those used here wouldn't normally work. Of course, a careful modification of the filters, the dimensions and some other things among the codes can turn these codes into useful tools to manage faster, easier and safer the raw data.

It also provides a more clear explanation of what is going on behind the theory of the two-stage DEA model in contrast with the difficult interpretation of the original paper finally finds some interesting results about the impact of the advertising and communications expense on the inefficiency of the banks.

6.2. Importance

The importance of this thesis is not more than two things. The first one is the construction of the algorithms that for the first time (at least as the search on the internet showed) were constructed and given for free as a full function. And secondly, the results about the effect of the banks' spending on their technical efficiency that show an interesting story on the banking efficiency field.

7. Appendix

7.1. Description of the distance function

We can now express $\boldsymbol{y}$ in terms of its polar coordinates turning $\boldsymbol{y_i}$ into $(\boldsymbol{\eta_i}, \boldsymbol{\delta_i})$. To do that, the angle and the distance are needed. Representing the angle of the $\boldsymbol{i^{th}}$ DMU corresponding to the $\boldsymbol{j^{th}}$ output variable with $\boldsymbol{\eta_{i,j}}$ and the distance with $\boldsymbol{\delta_i}$ as before, we calculating the angle as

$$\eta_{ij} = \begin{cases} \arctan(y_{i,j+1}/y_{i,1}), & for\ y_{i,1} > 0 \\ \pi/2, & if\ y_{i,1} = 0 \end{cases}$$

For the calculation of distance $\boldsymbol{\delta_i}$, though it's just the vertical distance from the point to the boundary (for technical efficiency) as reported before, is hard to be found like this for more dimensions. So we set $\boldsymbol{\omega(y_i) = \sqrt{y_i' y_i}}$ and we calculate the distance with the following formula that is identical to the Farrell efficient measure we demonstrated before (4).

$$\delta_i = \frac{\omega(\delta(x_i, y_i|P)y_i)}{\omega(y_i)} \qquad \text{(a1)}$$

Proof:

$$\delta_i = \frac{\omega(\delta(x_i, y_i|P)y_i)}{\omega(y_i)} = \delta(x_i, y_i|P)$$

$\delta_i = \frac{\omega(\delta(x_i,y_i|P)y_i)}{\omega(y_i)}$ => replacing $\omega(y_i)$ with $\sqrt{y_i' y_i}$

$=> \frac{\sqrt{(\delta(x_i,y_i|P)y_i)'\ \delta(x_i,y_i|P)y_i}}{\sqrt{y_i' y_i}}$ => as $\delta(x_i, y_i|P)$ is the Farrell formula for the distance and distance is a scalar

$$=> \frac{\sqrt{\delta(x_i,y_i|P)y_i'\ \delta(x_i,y_i|P)y_i}}{\sqrt{y_i' y_i}} = \frac{\sqrt{\delta(x_i,y_i|P)^2\ y_i' y_i}}{\sqrt{y_i' y_i}} = \frac{\delta(x_i,y_i|P)\sqrt{y_i' y_i}}{\sqrt{y_i' y_i}} = \delta(x_i, y_i|P)$$

For $\boldsymbol{j = 1, \dots, q-1}$ and $\boldsymbol{i = 1, \dots, n}$ where $\boldsymbol{n}$ is the total number of DMUs and $\boldsymbol{y_0 = [y_{i,1} \dots y_{i,q}]}$. Now we can write $\boldsymbol{y_i}$ as $(\boldsymbol{\eta_i}, \boldsymbol{\delta_i})$ where the angle $\boldsymbol{\eta_i = [\eta_{i,1}, \dots, \eta_{i,q-1}]}$.

7.2. Draws from a left-truncated normal distribution

The error term u_i with iid draws from a left-truncated $N(0, \sigma^2)$ distribution, truncated at a constant c can easily be constructed following the steps suggested by Simar and Wilson (2007):

1) Compute the constant c as described in the algorithm.
2) Compute $c' = c/\sigma$.
3) Generate v of length n to be uniform on $(0, 1)$.
4) Set $v' = \Phi(c') + [1 - \Phi(c')]v$ where Φ denotes the standard normal distribution function.
5) Finally, compute the error term as $u = \sigma\Phi^{-1}(v')$ where Φ^{-1} denotes the standard normal quantile function.

The estimated u is the error term drawn from the $N(0, \sigma^2)$ distribution with left-truncation at c.

7.3. Estimations in the algorithms

The Algorithm_1 and Algorithm_2 can be used for the estimation of confidence intervals for other statistics rather than the coefficients and the standard deviation. These statistics are sigma, R^2, AIC, BIC and RMSE. The sigma is just one of the coefficients automatically estimated by the truncated regression. The other statistics are programmed to compute as follows:

For some reason, there is not R^2 for truncated regressions in the whole literature and its not even given by any statistical software. So, if the user ask for an estimation of R^2, a pseudo-R^2 is constructed following the properties of a general accepted form of R^2 as the squared correlation between the fitted values of Y ($\hat{Y}$) and the actual Y as: $\mathbf{cor}(Y, \hat{Y})^2$. The same way as it would be in an OLS method.

$$R^2 = \mathbf{cor}(Y, \hat{Y})^2$$

If the user asked for the AIC (Akaike Information Criterion) it's constructed as:

$$AIC = -2 * logLik + 2 * m$$

Where **logLik** is the log Likelihood of the estimated truncated regression model and m is equal to **(k+1)**, where k is the number of parameters estimated (constant + slopes) and the +1 part is because the maximum likelihood method actually estimates the error term too (so the error term is an estimator of the model too). (In an OLS model the AIC would be the same but replacing m with k).

Almost the same happens with BIC (Bayesian Information Criterion or Schwarz Information Criterion). The formulas is:

$$BIC = -2 * logLik + \log(n) * m$$

Where 2 is replaced by the log of the number of observations.
Functions for **AIC** and **BIC** there are already in the basic R but it doesn't take care of the special case mentioned before about the truncated regression, so it was created from the scratch inside the algorithm as described.

Finally, the RMSE (Root Mean Square Error) is computed as:

$$RMSE = \sqrt{\frac{\sum e^2}{n}}.$$

7.4. Explanation of Algorithm_1_simplified

[1] Defines the algorithm "Algorithm_1_simplified"

[3-11] Checks if the needed package is already installed in the computer,
if not, it installs it.
Then it loads the package.
This sequence happens for both the package "Benchmarking" and "trunreg".

[13] Sets a seed for re-productivity reasons.

[15-17] Sets the class of the variables x, y and z as matrix in case that they are not because some of the following external functions cannot cooperate with objects of different class (e.g. data.frame).

[19] Checks if the matrix z contains column names. If not, then it attaches names to every variable it contains such as (z1, z2, ... zr). This is because it causes troubles on later commands when the algorithm tries to extract the names of the variables and finds nothing.

[19-22] It keeps asking a valid value for alpha it the user has mistakenly gave a value >=1 or <=0.
Finally, it saves the correct alpha (even if it was correct at the first time) as numeric because some later calculation can't be done with "character" vectors that may be mistakenly saved as by default.

[25-26] It defines the DEA model. The input data are x and y, RTS stands for Returns To Scale and it may be "CRS" (Constant Returns to Scale) or "VRS" (Variable Returns to Scale) with the "CRS" to be the default, the ORIENTATION may be "in" (input-oriented) or "out" (output-oriented) with the "out" to be the default.
Then saves the estimated efficiency scores (if ORIENTATION="in") or inefficiency scores (if ORIENTATION="out") at a variable named d_hat.

[29-47] The truncated regression part [2] in (Simar and Wilson, 2007).

[29-31] Sets the number of environmental variables at an object named r and defines two vectors that will be used later.

[33-41] Checks if the model is output or input oriented and creates the object dt which has the values of d_hat except for the observations where d_hat was <=1 where it replaces these observations with NA. The same check goes for the input oriented case with the only exception of d_hat >=1 instead of d_hat<=1.
This is a necessary step in order to run a truncated regression.
Then the appropriate truncated regression model is computed and saved in model_d_hat.

[43] The cells where dt has NAs are saved in the object bad.

[44] Only the values of dt that are not NAs are stores in non_na_dt.

[45] Because the truncated regression method has not an attribute for the residuals, the residuals are estimated as Y-Y^ and stored in trunk_resid.

[46-47] The standard deviation of the residuals are saved in sd_hat and the constant term and the slopes of the regression are saved in beta_hat.

[50-100] This is the step [3] as referred in (Simar and Wilson, 2007), although the actual loop covers rows [69-99].

[50] It creates the matrix that will contain the estimated bootstrapped values of β, this matrix is bootstrap_beta and its dimensions are [L,r+1].

[51] It defines the vectors bootstrap_sd that contains the bootstrapped value of σ.

[52] It sets the number of the inefficiency scores that are >1 (or the number of the efficiency scores that are <1) in the object m.

[53-56] It defines the objects c, zb, d_star and beta_hat_star as vectors.

[57] It defines the matrix z_m_dt that contains the values of z which their corresponding d_hat value is not NA. Thus, z_m_dt is the z matrix that contains only m observations instead of the initial n observations.

[59-65] A loop that creates the point of truncation (c) that is needed in step [3.1] and the object zb that contain the values $\sum_{j=1}^{m} \beta_j z_{i,j}$.

[67] It creates the object cc that contains the element that in (Simar and Wilson, 2007) is referred as c'.

[69-99] The actual bootstrap loop.

[71-73] The remaining steps for the left-truncated draws and the creating of the error term u.

[71] Creates the object v where it represents the uniform distribution on (0,1) with m elements.

[72] Creates the object vv where it represents what in (Simar and Wilson, 2007) is referred as v' and it is also of length m.

[73] Creates the desired error term u where each of its elements was drawn from N(0,sd_hat) with left-truncation at $c_i = 1 - \sum_{j=1}^{r1} \beta_j z_{i,j}$.

[76] It computes the d* as step [3.2] describes (zb+u) and stores it in the object d_star.

[79-93] This is the last step of the bootstrap loop [3.3] that estimates the bootstrapped statistics in every iteration.

[79-87] Checks if the model is output or input oriented and creates the object d_star_t which has the values of d_star except for the observations where d_star was <=1 where it replaces these observations with NA. The same check goes for the input oriented case with

the only exception of d_star >=1 instead of d_star<=1.
This is a necessary step in order to run a truncated regression.
Then the appropriate truncated regression model is computed and saved in trunk_d_star.

[89] The cells where d_star_t has NAs are saved in the object bad.

[90] Only the values of d_star_t that are not NAs are stores in non_na_d_star_t.

[91] Because the truncated regression method has not an attribute for the residuals, the residuals are estimated as Y-Y^ and stored in trunk_resid_2.

[92-93] The standard deviation of the residuals are saved in sd_hat_star and the constant term and the slopes of the regression are saved in beta_hat_star.

[95-97] Stores each estimated bootstrapped value of beta_hat_star at the matrix bootstrap_beta.

[98] it stores the estimated bootstrap values of sd_hat_star in the vector bootstrap_sd.

[100] It binds the objects bootstrap_beta and bootstrap_sd into a matrix A of length [L, r+2] where L is the number of iterations and r is the number of the external variables. The two extra columns are for the constant term and the standard deviation.

[103-152] Creates confidence intervals for the statistics.

[103] Defines the object quantiles for the storage of the bootstrap distribution quantiles. Its dimensions are [r+2, 2] where (r+2) is the number of columns of A.

[105-109] A loop that stores the lower and the upper quantiles of the bootstrap distribution of each statistic to object quantiles.
The lower quantile is the $alpha/2$ element of the sorted bootstrap distribution of each statistic and the upper quantile is the $1 - alpha/2$ element of the sorted bootstrap distribution of each statistic.

[110-111] Creates the objects b_ci and p_ci with dimensions [r+2, 2], where (r+2) is the number of columns of A.

[112] Defines the object statistic as a vector.

[115-132] The Basic Confidence Intervals

[116-119] Inserts the values (2*beta_hat – Upper quantile) and (2*beta_hat – Lower quantile) in b_ci that represent the lower and the upper confidence intervals respectively. Where, beta_hat is the initial statistic.

[120-121] Inserts the values (2*sd_hat – Upper quantile) and (2*sd_hat – Lower quantile) in b_ci that represent the lower and the upper confidence intervals respectively. Where, sd_hat is the initial statistic.

[123] Turns the class of b_ci into data.frame.

[125-126] Sets the name of each β in the object statistic (for example: $\beta_0, \beta_1, \beta_2$) and then binds it to the object b_ci.

[127] Prints the results (Confidence Intervals) on the screen with precision of 5 decimals.

[129-130] Sets the name of σ "sd" in the object statistic and then binds it to the object b_ci.

[131] Prints the results (Confidence Intervals) on the screen with precision of 5 decimals.

[135-152] The Percentile Confidence Intervals

[136-139] Inserts the Lower and the Upper quantiles of the sorted bootstrap distribution of each β in p_ci that represent the lower and the upper confidence intervals respectively.

[140-141] Inserts the Lower and the Upper quantiles of the sorted bootstrap distribution of σ in p_ci that represent the lower and the upper confidence intervals respectively.

[143] Turns the class of p_ci into data.frame.

[145-146] Sets the name of each β in the object statistic (for example: $\beta_0, \beta_1, \beta_2$) and then binds it to the object p_ci.

[147] Prints the results (Confidence Intervals) on the screen with precision of 5 decimals.

[149-150] Sets the name of σ "sd" in the object statistic and then binds it to the object p_ci.

[151] Prints the results (Confidence Intervals) on the screen with precision of 5 decimals.

7.5. Explanation of Algorithm_2_simplified

[1] Defines the algorithm "Algorithm_2_simplified"

[3-11] Checks if the needed package is already installed in the computer,
if not, it installs it.
Then it loads the package.
This sequence happens for both the package "Benchmarking" and "trunreg".

[13] Sets a seed for re-productivity reasons.

[15-17] Sets the class of the variables x, y and z as matrix in case that they are not because some of the following external functions cannot cooperate with objects of different class (e.g. data.frame).

[19] Checks if the matrix z contains column names. If not, then it attaches names to every variable it contains such as (z1, z2, ... zr). This is because it causes troubles on later commands when the algorithm tries to extract the names of the variables and finds nothing.

[19-22] It keeps asking a valid value for alpha it the user has mistakenly gave a value >=1 or <=0.
Finally, it saves the correct alpha (even if it was correct at the first time) as numeric because some later calculation can't be done with "character" vectors that may be mistakenly saved as by default.

[25-26] It defines the DEA model. The input data are x and y, RTS stands for Returns To Scale and it may be "CRS" (Constant Returns to Scale) or "VRS" (Variable Returns to Scale) with the "CRS" to be the default, the ORIENTATION may be "in" (input-oriented) or "out" (output-oriented) with the "out" to be the default.
Then saves the estimated efficiency scores (if ORIENTATION="in") or inefficiency scores (if ORIENTATION="out") at a variable named d_hat.

[29-47] The truncated regression part [2] in (Simar and Wilson, 2007).

[29-31] Sets the number of environmental variables at an object named r and defines two vectors that will be used later.

[33-41] Checks if the model is output or input oriented and creates the object d_hat_t which has the values of d_hat except for the observations where d_hat was <=1 where it replaces these observations with NA. The same check goes for the input oriented case with the only exception of d_hat >=1 instead of d_hat<=1.
This is a necessary step in order to run a truncated regression.
Then the appropriate truncated regression model is computed and saved in model_d_hat.

[43] The cells where d_hat_t has NAs are saved in the object bad.

[44] Only the values of d_hat_t that are not NAs are stores in non_na_d_hat_t.

[45] Because the truncated regression method has not an attribute for the residuals, the residuals are estimated as Y-Y^ and stored in trunk_resid.

[46-47] The standard deviation of the residuals are saved in sd_hat and the slopes of the regression are saved in beta_hat.

[50-86] This is the step [3] as referred in (Simar and Wilson, 2007), although the actual loop covers rows [66-86].

[50-53] It defines the objects c, zb and d_star as vectors and sets the object n equal to the number of observations in z.

[56-62] A loop that creates the point of truncation (c) that is needed in step [3.1] and the object zb that contain the values $\sum_{j=1}^{n} \beta_j z_{i,j}$.

[64] It creates the object cc that contains the element that in (Simar and Wilson, 2007) is referred as c'.

[65] Defines the matrix B with dimensions [n, L1] where L1 is the number of iteration of the first bootstrap.

[66-86] The actual bootstrap loop.

[68-70] The remaining steps for the left-truncated draws and the creating of the error term u.

[68] Creates the object v where it represents the uniform distribution on (0,1) with m elements.

[69] Creates the object vv where it represents what in (Simar and Wilson, 2007) is referred as v' and it is also of length m.

[70] Creates the desired error term u where each of its elements was drawn from N(0,sd_hat) with left-truncation at $c_i = 1 - \sum_{j=1}^{n} \beta_j z_{i,j}$.

[73] It computes the d* as step [3.2] describes (zb+u) and stores it in the object d_star.

[76-77] Constructs the new objects x_star and y_star.

[80-81] It defines the second DEA model. The input data are x_star and y_star, RTS stands for Returns To Scale and it may be "CRS" (Constant Returns to Scale) or "VRS" (Variable Returns to Scale) with the "CRS" to be the default, the ORIENTATION may be "in" (input-oriented) or "out" (output-oriented) with the "out" to be the default.
Then saves the estimated efficiency scores (if ORIENTATION="in") or inefficiency scores (if ORIENTATION="out") at a variable named d_hat_star.

[83-85] Adds every estimated set of inefficiencies in columns of the matrix B.

[88] Defines the object mean_di_hat_star as a vector.

[90-92] Stores the mean of every set of inefficiency scores into the object mean_di_hat_star.

[94] Computes the bias for each observation of the inefficiency scores and stores them in the object bias.

[95] Computes the bias-corrected $\hat{\delta}$ and stores it in the object d_2hat.

[98-114] This is the last step of the bootstrap loop [5] that estimates the bootstrapped statistics in every iteration.

[98] Defines the object beta_2hat as a vector.

[100-108] Checks if the model is output or input oriented and creates the object d_2hat_t which has the values of d_2hat except for the observations where d_2hat was <=1 where it replaces these observations with NA. The same check goes for the input oriented case with the only exception of d_2hat >=1 instead of d_2hat<=1.
This is a necessary step in order to run a truncated regression.
Then the appropriate truncated regression model is computed and saved in model_d_2hat.

[110] The cells where d_2hat_t has NAs are saved in the object bad.

[111] Only the values of d_2hat_t that are not NAs are stores in non_na_d_2hat_t.

[112] Because the truncated regression method has not an attribute for the residuals, the residuals are estimated as Y-Y^ and stored in trunk_resid.

[113-114] The standard deviation of the residuals are saved in sd_2hat and the constant term and the slopes of the regression are saved in beta_2hat.

[117-164] This is the step [6] as referred in (Simar and Wilson, 2007), although the actual loop covers rows [134-164].

[117] It defines the matrix that will contain the estimated bootstrapped values of β, this matrix is bootstrap_beta and its dimensions are [L2,r+1].

[118] It defines the vector bootstrap_sd that contains the bootstrapped value of σ.

[119-122] It defines the objects c_2, zb_2, d_2star and beta_2hat_star as vectors.

[124-130] A loop that creates the point of truncation c_2 that is needed in step [6.1] and the object zb_2 that contain the values $\sum_{j=1}^{n} \beta_j z_{i,j}$.

[132] It creates the object cc_2 that contains the element that in (Simar and Wilson, 2007) is referred as c'.

[134-164] The actual bootstrap loop.

[136-138] The remaining steps for the left-truncated draws and the creation of the error term u.

[136] Creates the object v_2 where it represents the uniform distribution on (0,1) with n elements.

[137] Creates the object vv_2 where it represents what in (Simar and Wilson, 2007) is referred as v' and it is also of length n.

[138] Creates the desired error term u where each of its elements was drawn from N(0,sd_2hat) with left-truncation at $c_i = 1 - \sum_{j=1}^{n} \beta_j z_{i,j}$ and it's stored in the object u_2.

[141] It computes the d** as step [6.2] describes (zb_2+u_2) and stores it in the object d_2star.

[144-158] This is the last step of the bootstrap loop [6.3] that estimates the bootstrapped statistics in every iteration.

[144-152] Checks if the model is output or input oriented and creates the object d_2star_t which has the values of d_2star except for the observations where d_2star was <=1 where it replaces these observations with NA. The same check goes for the input oriented case with the only exception of d_2star >=1 instead of d_2star<=1.
This is a necessary step in order to run a truncated regression.
Then the appropriate truncated regression model is computed and saved in model_d_2star.

[154] The cells where d_2star_t has NAs are saved in the object bad.

[155] Only the values of d_2star_t that are not NAs are stores in non_na_d_2star_t.

[156] Because the truncated regression method has not an attribute for the residuals, the residuals are estimated as Y-Y^ and stored in trunk_resid.

[157-158] The standard deviation of the residuals are saved in sd_2hat_star and the constant term and the slopes of the regression are saved in beta_2hat_star.

[160-162] Stores each estimated bootstrapped value of beta_2hat_star at the matrix bootstrap_beta.

[163] it stores the estimated bootstrap values of sd_2hat_star in the vector bootstrap_sd.

[165] It binds the objects bootstrap_beta and bootstrap_sd into a matrix C of length [L2, r+2] where L2 is the number of iterations and r is the number of the external variables. The two extra columns are for the constant term and the standard deviation.

[168-217] Creates confidence intervals for the statistics.

[168] Defines the object quantiles for the storage of the bootstrap distribution quantiles. Its dimensions are [r+2, 2] where (r+2) is the number of columns of C.

[170-174] A loop that stores the lower and the upper quantiles of the bootstrap distribution of each statistic to object quantiles.
The lower quantile is the $alpha/2$ element of the sorted bootstrap distribution of each statistic and the upper quantile is the $1 - alpha/2$ element of the sorted bootstrap distribution of each statistic.

[175-176] Creates the objects b_ci and p_ci with dimensions [r+2, 2], where (r+2) is the number of columns of C.

[177] Defines the object statistic as a vector.

[180-197] The Basic Confidence Intervals

[181-184] Inserts the values (2*beta_2hat – Upper quantile) and (2*beta_2hat – Lower quantile) in b_ci that represent the lower and the upper confidence intervals respectively. Where, beta_2hat is the initial statistic.

[185-186] Inserts the values (2*sd_2hat – Upper quantile) and (2*sd_2hat – Lower quantile) in b_ci that represent the lower and the upper confidence intervals respectively. Where, sd_2hat is the initial statistic.

[188] Turns the class of b_ci into data.frame.

[190-191] Sets the name of each β in the object statistic (for example: $\beta_0, \beta_1, \beta_2$) and then binds it to the object b_ci.

[192] Prints the results (Confidence Intervals) on the screen with precision of 5 decimals.

[194-195] Sets the name of σ "sd" in the object statistic and then binds it to the object b_ci.

[196] Prints the results (Confidence Intervals) on the screen with precision of 5 decimals.

[200-217] The Percentile Confidence Intervals

[201-204] Inserts the Lower and the Upper quantiles of the sorted bootstrap distribution of each β in p_ci that represent the lower and the upper confidence intervals respectively.

[205-206] Inserts the Lower and the Upper quantiles of the sorted bootstrap distribution of σ in p_ci that represent the lower and the upper confidence intervals respectively.

[208] Turns the class of p_ci into data.frame.

[210-211] Sets the name of each β in the object statistic (for example: $\beta_0, \beta_1, \beta_2$) and then binds it to the object p_ci.

[212] Prints the results (Confidence Intervals) on the screen with precision of 5 decimals.

[214-215] Sets the name of σ "sd" in the object statistic and then binds it to the object p_ci.

[216] Prints the results (Confidence Intervals) on the screen with precision of 5 decimals.

Algorithm_1(x, y, z, RTS="CRS", ORIENTATION="out", l=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.05662	-0.42619	0.46357	X	-0.55861	0.33114	X
2	Advertising.Expense	0.00091	7e-04	0.00115	I	0.00071	0.00116	I
3	Communications.Expense	-7e-05	-0.00013	-2e-05	I	-0.00013	-1e-05	I
4	sigma	0.18289	0.14144	0.22449		0.56902	0.65207	
5	P-value(b0)	0.45419	8e-05	0.97406		0.81153	1.78551	
6	P-value(b1)	0	0	0	***	0	0	***
7	P-value(b2)	0.08024	0	0.57699	*	-0.00392	0.57307	*
8	P-value(b3)	0	0	0	***	0	0	***
9	sd	0.1349	0.08808	0.19113		0.18772	0.29077	
10	R^2	0.77538	0.7275	0.81471		-0.18458	-0.09737	
11	AIC	-5452.28957	-5580.71166	-5323.11863		392.07549	649.66852	
12	BIC	-5432.0511	-5560.47319	-5302.88016		412.31396	669.90699	
13	RMSE	0.95548	0.5535	1.43004		0.8774	1.75395	

Algorithm_2(x, y, z, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.01836	-0.67038	0.57311	X	-40.20585	-38.96236	!
2	Advertising.Expense	0.01027	0.0093	0.01124	!	0.00935	0.01129	!
3	Communications.Expense	-0.00124	-0.00181	-0.00066	!	-0.00184	-0.00068	!
4	sigma	1.96358	1.76488	2.19404		2.15743	2.58659	
5	P-value(b0)	0.49351	0.01287	0.97644		-0.45508	0.50849	
6	P-value(b1)	0	0	0	***	0.36086	0.36086	***
7	P-value(b2)	0.00298	0	0.02856	***	0.51056	0.53912	***
8	P-value(b3)	0	0	0	***	0.03185	0.03185	***
9	sd	1.16271	1.09146	1.24346		2.70713	2.85914	
10	R^2	0.76313	0.73531	0.78896		-0.50745	-0.4538	
11	AIC	2975.56241	2873.02592	3081.31199		-5191.96742	-4983.68135	
12	BIC	2995.86903	2893.33254	3101.61861		-5172.09109	-4963.80502	
13	RMSE	2.40475	1.92019	3.02293		38.44132	39.54406	

Algorithm_1(x, y, lnz, RTS="CRS", ORIENTATION="out", L=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	-0.05105	-1.29152	0.52601	X	0.55018	2.36772	!
2	ln(Advertising.Expense)	0.10781	0.04403	0.23435	!	-0.02627	0.16405	X
3	ln(Communications.Expense)	0.04853	0.0055	0.11333	!	-0.01886	0.08897	X
4	sigma	0.21567	0.15303	0.32018		0.16507	0.33223	
5	P-value(b0)	0.50979	0.00014	0.97319		-0.97319	-0.00014	
6	P-value(b1)	0.0249	0.00027	0.12875	**	-0.12875	-0.00027	**
7	P-value(b2)	0.13151	0.00251	0.73045		-0.72962	-0.00168	
8	P-value(b3)	0.00036	0	0.00183	***	-0.00183	0	***
9	sd	0.26472	0.13858	0.53952		-0.10943	0.29151	
10	R^2	0.20157	0.14633	0.2599		0.53593	0.6495	
11	AIC	-4643.79698	-4769.89494	-4519.91188		-967.23678	-717.25371	
12	BIC	-4623.55851	-4749.65647	-4499.67341		-946.99831	-697.01524	
13	RMSE	1.00051	0.47015	2.17462		-1.15802	0.54645	

Algorithm_2(x, y, lnz, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.00301	-0.38152	0.32428	X	-3.20841	-2.50261	I
2	ln(Advertising.Expense)	0.3992	0.31682	0.4917	I	0.31019	0.48506	I
3	ln(Communications.Expense)	0.08922	0.02086	0.16018	I	0.01648	0.1558	I
4	sigma	0.80727	0.72202	0.90237		0.42839	0.60875	
5	P-value(b0)	0.50086	0.01561	0.97073		-0.93669	0.01842	
6	P-value(b1)	0	0	0	***	1e-05	1e-05	***
7	P-value(b2)	0.07437	1e-05	0.55281	*	-0.3968	0.156	*
8	P-value(b3)	0	0	0	***	0	0	***
9	sd	0.69465	0.59847	0.81101		0.80934	1.02188	
10	R^2	0.437	0.38953	0.48179		0.16683	0.25909	
11	AIC	541.38728	432.89534	653.0114		-3443.97583	-3223.85977	
12	BIC	561.69389	453.20196	673.31802		-3424.17832	-3204.06226	
13	RMSE	1.34833	1.06441	1.69432		3.06416	3.69407	

Algorithm_1(x, y, lnz21, RTS="CRS", ORIENTATION="out", L=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=5)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	-0.08785	-2.00941	0.5839	X	0.68439	3.2777	!
2	ln(Advertising.Expense)	0.07573	0.01559	0.23117	!	-0.09138	0.1242	X
3	ln(Advertising.Expense)^2	0.00628	-0.00162	0.01666	X	-0.00411	0.01417	X
4	ln(Communications.Expense)	0.03208	-0.01533	0.11192	X	-0.05349	0.07376	X
5	sigma	0.1778	0.11748	0.30064		0.14986	0.33302	
6	P-value(b0)	0.50357	1e-05	0.9773		-0.9773	-1e-05	
7	P-value(b1)	0.14457	0.01503	0.43784		-0.43776	-0.01496	
8	P-value(b2)	0.2175	0.00199	0.90898		-0.89665	0.01035	
9	P-value(b3)	0.36989	0.0361	0.95793		-0.90545	0.01638	
10	P-value(b4)	0.00251	0	0.03366	***	-0.03366	0	***
11	sd	0.23824	0.10412	0.60655		-0.25349	0.24894	
12	R^2	0.21761	0.15059	0.2904		0.60185	0.74166	
13	AIC	-5540.01269	-5670.47877	-5414.80514		-81.079	174.59463	
14	BIC	-5514.7146	-5645.18068	-5389.50705		-55.78091	199.89272	
15	RMSE	1.02716	0.40763	2.80705		-1.96385	0.43557	

Algorithm_2(x, y, lnz21, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.00585	-0.43601	0.32797	X	-1.42543	-0.66146	I
2	ln(Advertising.Expense)	0.2578	0.17385	0.36443	I	0.15444	0.34503	I
3	ln(Advertising.Expense)^2	0.00692	-0.00506	0.01827	X	-0.00432	0.01901	X
4	ln(Communications.Expense)	0.07281	0.01649	0.13566	I	0.00837	0.12754	I
5	sigma	0.55633	0.48746	0.6369		0.49095	0.64039	
6	P-value(b0)	0.48996	0.01179	0.9788		-0.72657	0.24043	
7	P-value(b1)	0	0	1e-05	***	0.00026	0.00027	***
8	P-value(b2)	0.32487	0.00065	0.94742		-0.2683	0.67848	
9	P-value(b3)	0.08506	7e-05	0.56204	*	-0.39861	0.16336	*
10	P-value(b4)	0	0	0	***	0	0	***
11	sd	0.50095	0.39685	0.64453		0.47485	0.72252	
12	R^2	0.46703	0.40671	0.52782		0.20832	0.32943	
13	AIC	-732.98544	-847.42021	-616.27812		-2001.64311	-1770.50101	
14	BIC	-707.60217	-822.03694	-590.89485		-1976.7178	-1745.57571	
15	RMSE	1.13081	0.83755	1.52682		1.60539	2.29465	

Algorithm_1(x, y, lnz12, RTS="CRS", ORIENTATION="out", L=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	-0.03404	-1.14719	0.50451	X	0.40466	2.05636	!
2	ln(Advertising.Expense)	0.11611	0.05321	0.22847	!	-0.00063	0.17462	X
3	ln(Communications.Expense)	0.09737	0.01559	0.2318	!	-0.04609	0.17011	X
4	ln(Communications.Expense)^2	-0.00725	-0.02189	0.00196	X	-0.01542	0.00843	X
5	sigma	0.24063	0.17806	0.34166		0.16318	0.32679	
6	P-value(b0)	0.49703	0.00034	0.97141		-0.97133	-0.00027	
7	P-value(b1)	0.01173	6e-05	0.06608	**	-0.06608	-6e-05	**
8	P-value(b2)	0.11403	0.00424	0.56563		-0.56404	-0.00266	
9	P-value(b3)	0.28953	0.01591	0.91658		-0.82415	0.07652	
10	P-value(b4)	1e-04	0	2e-04	***	-2e-04	0	***
11	sd	0.28514	0.15809	0.54288		-0.06043	0.32435	
12	R^2	0.20508	0.15207	0.26104		0.49772	0.60669	
13	AIC	-4081.5456	-4206.40307	-3958.30014		-1534.17811	-1286.07518	
14	BIC	-4056.24751	-4181.10498	-3933.00205		-1508.88002	-1260.77709	
15	RMSE	0.97214	0.49765	1.94241		-0.81264	0.63212	

Algorithm_2(x, y, lnz12, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.0117	-0.49809	0.44346	X	-7.20457	-6.26301	!
2	ln(Advertising.Expense)	0.62248	0.50509	0.74737	!	0.5046	0.74687	!
3	ln(Communications.Expense)	0.45454	0.26432	0.65873	!	0.2558	0.65022	!
4	ln(Communications.Expense)^2	-0.05016	-0.07835	-0.02222	!	-0.07935	-0.02323	!
5	sigma	1.35983	1.23543	1.49713		0.15115	0.41285	
6	P-value(b0)	0.49546	0.0138	0.97826		-0.93873	0.02573	
7	P-value(b1)	0	0	0	***	0.00147	0.00147	***
8	P-value(b2)	8e-04	0	0.00594	***	0.02793	0.03387	***
9	P-value(b3)	0.00918	0	0.09232	***	-0.0428	0.04953	***
10	P-value(b4)	0	0	0	***	0	0	***
11	sd	1.13419	1.00419	1.28762		1.44503	1.72847	
12	R^2	0.41918	0.37378	0.46457		0.09925	0.19004	
13	AIC	2273.55295	2172.70306	2377.21061		-5286.87435	-5082.3668	
14	BIC	2298.93622	2198.08633	2402.59388		-5262.36427	-5057.85672	
15	RMSE	1.78823	1.47586	2.17836		5.79084	6.49334	

Algorithm_1(x, y, lnz22, RTS="CRS", ORIENTATION="out", L=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	-0.02347	-1.23606	0.55952	X	0.62823	2.42381	!
2	ln(Advertising.Expense)	0.02748	-0.0208	0.10524	X	-0.0523	0.07374	X
3	ln(Advertising.Expense)^2	0.01946	0.00867	0.03898	!	4e-05	0.03034	!
4	ln(Communications.Expense)	0.13587	0.04465	0.3217	!	-0.05809	0.21896	X
5	ln(Communications.Expense)^2	-0.02136	-0.04584	-0.00793	!	-0.03393	0.00398	X
6	sigma	0.17015	0.11786	0.25507		0.1805	0.31771	
7	P-value(b0)	0.49846	2e-05	0.97791		-0.97791	-2e-05	
8	P-value(b1)	0.41401	0.05673	0.96128		-0.81994	0.08462	
9	P-value(b2)	0.02077	6e-05	0.12321	**	-0.12321	-6e-05	**
10	P-value(b3)	0.04828	0.00155	0.20344	**	-0.20344	-0.00155	**
11	P-value(b4)	0.03716	0.00058	0.17698	**	-0.17698	-0.00058	**
12	P-value(b5)	3e-04	0	0.00307	***	-0.00307	0	***
13	sd	0.2236	0.10767	0.45959		-0.11496	0.23696	
14	R^2	0.23364	0.16933	0.29671		0.6375	0.76488	
15	AIC	-5490.95631	-5618.29752	-5364.90442		-192.67021	60.7229	
16	BIC	-5460.5986	-5587.93982	-5334.54671		-162.3125	91.08061	
17	RMSE	0.9066	0.39927	1.97148		-1.17057	0.40164	

Algorithm_2(x, y, lnz22, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.01239	-0.35636	0.29537	X	-1.333	-0.68127	!
2	ln(Advertising.Expense)	0.1005	0.03707	0.17495	!	0.02582	0.1637	!
3	ln(Advertising.Expense)^2	0.04346	0.03	0.05774	!	0.03038	0.05812	!
4	ln(Communications.Expense)	0.34947	0.24677	0.46165	!	0.24521	0.4601	!
5	ln(Communications.Expense)^2	-0.05396	-0.07162	-0.03714	!	-0.07244	-0.03795	!
6	sigma	0.49454	0.43777	0.55956		0.49333	0.61511	
7	P-value(b0)	0.4895	0.00885	0.97386		-0.78626	0.17875	
8	P-value(b1)	0.02736	3e-05	0.22311	**	-0.1313	0.09178	**
9	P-value(b2)	1e-05	0	3e-05	***	-2e-05	1e-05	***
10	P-value(b3)	0	0	0	***	1e-05	1e-05	***
11	P-value(b4)	0	0	1e-05	***	0	1e-05	***
12	P-value(b5)	0	0	0	***	0	0	***
13	sd	0.42892	0.35324	0.52624		0.47056	0.64357	
14	R^2	0.53347	0.4834	0.58291		0.22499	0.3245	
15	AIC	-900.7409	-1015.13059	-783.98806		-1899.57627	-1668.43373	
16	BIC	-870.28097	-984.67067	-753.52813		-1869.67652	-1638.53398	
17	RMSE	0.9477	0.72047	1.24231		1.50558	2.02743	

Algorithm_1(x, y, lnz32, RTS="CRS", ORIENTATION="out", L=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.01641	-1.28356	0.58655	X	0.62661	2.49672	!
2	ln(Advertising.Expense)	0.02625	-0.02109	0.10217	X	-0.05017	0.07308	X
3	ln(Advertising.Expense)^2	0.01546	-0.00188	0.04387	X	-0.01039	0.03535	X
4	ln(Advertising.Expense)^3	0.00054	-0.00186	0.00258	X	-0.00169	0.00275	X
5	ln(Communications.Expense)	0.13769	0.04793	0.32123	!	-0.04769	0.22562	X
6	ln(Communications.Expense)^2	-0.02212	-0.04702	-0.00866	!	-0.03549	0.00288	X
7	sigma	0.16421	0.11269	0.25327		0.17768	0.31826	
8	P-value(b0)	0.47292	1e-05	0.97312		-0.97312	-1e-05	
9	P-value(b1)	0.41705	0.05761	0.96404		-0.81176	0.09467	
10	P-value(b2)	0.24914	0.01279	0.90709		-0.90437	-0.01008	
11	P-value(b3)	0.44773	0.01424	0.97168		0.0532	1.01064	
12	P-value(b4)	0.04334	0.00061	0.20354	**	-0.20354	-0.00061	**
13	P-value(b5)	0.03273	0.00015	0.1745	**	-0.1745	-0.00015	**
14	P-value(b6)	0.00044	0	0.00492	***	-0.00492	0	***
15	sd	0.21404	0.09987	0.46453		-0.12503	0.23962	
16	R^2	0.23541	0.16924	0.30145		0.63675	0.76896	
17	AIC	-5565.20814	-5692.89898	-5438.34048		-116.07797	138.48052	
18	BIC	-5529.79082	-5657.48165	-5402.92316		-80.66065	173.89785	
19	RMSE	0.87391	0.38225	2.02193		-1.24006	0.39962	

Algorithm_2(x, y, lnz32, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.02096	-0.34235	0.31376	X	-1.40634	-0.75023	I
2	ln(Advertising.Expense)	0.09497	0.03549	0.16633	I	0.02321	0.15405	I
3	ln(Advertising.Expense)^2	0.05581	0.03337	0.07866	I	0.0368	0.08209	I
4	ln(Advertising.Expense)^3	-0.00181	-0.00478	0.00123	X	-0.00523	0.00079	X
5	ln(Communications.Expense)	0.32497	0.21832	0.43976	I	0.21607	0.4375	I
6	ln(Communications.Expense)^2	-0.04914	-0.06841	-0.03084	I	-0.06847	-0.03091	I
7	sigma	0.49725	0.44008	0.56277		0.49885	0.62154	
8	P-value(b0)	0.48595	0.00653	0.97722		-0.79541	0.17527	
9	P-value(b1)	0.02892	3e-05	0.24471	**	-0.13951	0.10518	**
10	P-value(b2)	8e-04	0	0.00649	***	-0.00439	0.0021	***
11	P-value(b3)	0.3149	0.00237	0.9387		-0.27712	0.65921	
12	P-value(b4)	1e-05	0	5e-05	***	3e-05	8e-05	***
13	P-value(b5)	0.00013	0	0.00102	***	-0.00086	0.00016	***
14	P-value(b6)	0	0	0	***	0	0	***
15	sd	0.42668	0.35208	0.5203		0.48529	0.6535	
16	R^2	0.54344	0.49507	0.59106		0.22098	0.31698	
17	AIC	-857.22045	-970.56403	-742.16239		-1919.10663	-1690.70499	
18	BIC	-821.68388	-935.02745	-706.62581		-1884.22359	-1655.82194	
19	RMSE	0.93767	0.71492	1.22777		1.55693	2.06977	

Algorithm_1(x, y, lnz23, RTS="CRS", ORIENTATION="out", L=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	-0.00274	-1.23329	0.56901	X	0.61918	2.42148	!
2	ln(Advertising.Expense)	0.02654	-0.02065	0.10213	X	-0.0492	0.07358	X
3	ln(Advertising.Expense)^2	0.0192	0.00834	0.03941	!	-0.00043	0.03065	X
4	ln(Communications.Expense)	0.13045	-0.01167	0.3401	X	-0.07753	0.27425	X
5	ln(Communications.Expense)^2	-0.01979	-0.07448	0.02851	X	-0.06997	0.03302	X
6	ln(Communications.Expense)^3	-0.00011	-0.00409	0.00355	X	-0.00358	0.00405	X
7	sigma	0.16853	0.11727	0.25203		0.18354	0.3183	
8	P-value(b0)	0.48566	2e-05	0.97444		-0.97444	-2e-05	
9	P-value(b1)	0.4152	0.05323	0.96427		-0.82237	0.08868	
10	P-value(b2)	0.02299	8e-05	0.14073	**	-0.14073	-8e-05	**
11	P-value(b3)	0.23203	0.011	0.88215		-0.87679	-0.00564	
12	P-value(b4)	0.42483	0.03037	0.96749		-0.65773	0.27939	
13	P-value(b5)	0.51526	0.07069	0.97394		1.00321	1.90646	
14	P-value(b6)	0.00031	0	0.00364	***	-0.00364	0	***
15	sd	0.21998	0.10637	0.46604		-0.12149	0.23818	
16	R^2	0.23473	0.17081	0.30048		0.63391	0.76357	
17	AIC	-5491.53342	-5618.91093	-5364.47577		-189.09926	65.3359	
18	BIC	-5456.1161	-5583.49361	-5329.05845		-153.68194	100.75322	
19	RMSE	0.88962	0.39343	1.93926		-1.13843	0.4074	

Algorithm_2(x, y, lnz23, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.01711	-0.36278	0.30711	X	-1.34503	-0.67515	!
2	ln(Advertising.Expense)	0.09991	0.03634	0.17492	!	0.02578	0.16437	!
3	ln(Advertising.Expense)^2	0.04351	0.02957	0.05835	!	0.02986	0.05865	!
4	ln(Communications.Expense)	0.35144	0.17212	0.54955	!	0.16128	0.53871	!
5	ln(Communications.Expense)^2	-0.05493	-0.11996	0.00564	X	-0.11675	0.00885	X
6	ln(Communications.Expense)^3	9e-05	-0.00502	0.00523	X	-0.0051	0.00514	X
7	sigma	0.49346	0.43695	0.55889		0.49385	0.61579	
8	P-value(b0)	0.48243	0.00945	0.96653		-0.76905	0.18803	
9	P-value(b1)	0.02798	4e-05	0.22546	**	-0.13317	0.09225	**
10	P-value(b2)	1e-05	0	6e-05	***	-5e-05	2e-05	***
11	P-value(b3)	0.0071	0	0.05868	***	-0.04171	0.01696	***
12	P-value(b4)	0.19813	0.00036	0.88776		-0.49936	0.38805	
13	P-value(b5)	0.50808	0.02308	0.9793		0.99021	1.94642	
14	P-value(b6)	0	0	0	***	0	0	***
15	sd	0.42793	0.35347	0.53074		0.46613	0.6434	
16	R^2	0.53403	0.48061	0.58691		0.22083	0.32713	
17	AIC	-898.42856	-1012.34957	-784.23352		-1894.90618	-1666.79013	
18	BIC	-862.89198	-976.813	-748.69695		-1860.02314	-1631.90709	
19	RMSE	0.94326	0.71957	1.2445		1.50246	2.02738	

Algorithm_1(x, y, lnz33, RTS="CRS", ORIENTATION="out", L=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.04538	-1.28323	0.61071	X	0.62942	2.52336	!
2	ln(Advertising.Expense)	0.02501	-0.02177	0.09983	X	-0.04873	0.07286	X
3	ln(Advertising.Expense)^2	0.01251	-0.00814	0.04597	X	-0.01591	0.03821	X
4	ln(Advertising.Expense)^3	0.00084	-0.00169	0.00332	X	-0.00204	0.00297	X
5	ln(Communications.Expense)	0.11117	-0.03228	0.33897	X	-0.09843	0.27282	X
6	ln(Communications.Expense)^2	-0.0117	-0.07288	0.03874	X	-0.06934	0.04228	X
7	ln(Communications.Expense)^3	-0.00088	-0.00534	0.0033	X	-0.00449	0.00415	X
8	sigma	0.15952	0.10894	0.25018		0.17844	0.31968	
9	P-value(b0)	0.46097	0	0.97648		-0.97648	0	
10	P-value(b1)	0.4244	0.05999	0.97002		-0.8083	0.10174	
11	P-value(b2)	0.37598	0.0308	0.94937		-0.90974	0.00883	
12	P-value(b3)	0.40844	0.00796	0.96656		-0.11283	0.84577	
13	P-value(b4)	0.30237	0.02269	0.92567		-0.9118	-0.00882	
14	P-value(b5)	0.49489	0.0645	0.96851		-0.30647	0.59755	
15	P-value(b6)	0.48533	0.03498	0.97278		0.33929	1.27708	
16	P-value(b7)	0.00043	0	0.00509	***	-0.00509	0	***
17	sd	0.20726	0.09409	0.47354		-0.13926	0.24019	
18	R^2	0.23607	0.16937	0.30634		0.64001	0.77699	
19	AIC	-5653.65813	-5781.3503	-5525.6888		-25.12188	230.53962	
20	BIC	-5613.18119	-5740.87336	-5485.21186		15.35506	271.01656	
21	RMSE	0.85814	0.36559	2.03317		-1.26251	0.40508	

Algorithm_2(x, y, lnz33, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.03062	-0.35431	0.33696	X	-1.49071	-0.79944	!
2	ln(Advertising.Expense)	0.09797	0.03759	0.16838	!	0.0284	0.15919	!
3	ln(Advertising.Expense)^2	0.05901	0.03172	0.0865	!	0.03699	0.09177	!
4	ln(Advertising.Expense)^3	-0.00217	-0.00557	0.0015	X	-0.00642	0.00065	X
5	ln(Communications.Expense)	0.37839	0.194	0.59764	!	0.1795	0.58315	!
6	ln(Communications.Expense)^2	-0.07088	-0.14548	-0.00375	!	-0.14465	-0.00293	!
7	ln(Communications.Expense)^3	0.00189	-0.0039	0.00812	X	-0.00385	0.00817	X
8	sigma	0.5034	0.44572	0.57081		0.49358	0.61867	
9	P-value(b0)	0.47844	0.00677	0.97309		-0.78024	0.18609	
10	P-value(b1)	0.02456	1e-05	0.20143	**	-0.1152	0.08622	**
11	P-value(b2)	0.00352	0	0.02911	***	-0.02012	0.00898	***
12	P-value(b3)	0.30256	0.00179	0.92746		-0.29049	0.63519	
13	P-value(b4)	0.00495	0	0.04032	***	-0.02426	0.01606	***
14	P-value(b5)	0.14499	0.00018	0.81389		-0.55351	0.2602	
15	P-value(b6)	0.44458	0.011	0.97712		0.21007	1.17619	
16	P-value(b7)	0	0	0	***	0	0	***
17	sd	0.43172	0.35604	0.53067		0.49033	0.66497	
18	R^2	0.53947	0.48788	0.59044		0.21337	0.31594	
19	AIC	-764.02246	-876.52081	-649.74921		-1996.45138	-1769.67977	
20	BIC	-723.40923	-835.90758	-609.13597		-1956.55675	-1729.78514	
21	RMSE	0.92239	0.70263	1.21951		1.57753	2.09441	

Algorithm_1(x, y, lnz31, RTS="CRS", ORIENTATION="out", l=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	-0.04092	-1.68445	0.58446	X	0.59411	2.86301	I
2	ln(Advertising.Expense)	0.06566	0.01008	0.18853	I	-0.06574	0.11271	X
3	ln(Advertising.Expense)^2	0.01821	-0.00103	0.0527	X	-0.01535	0.03838	X
4	ln(Advertising.Expense)^3	-0.00151	-0.00509	0.00057	X	-0.00362	0.00204	X
5	ln(Communications.Expense)	0.03166	-0.01209	0.1042	X	-0.04454	0.07175	X
6	sigma	0.17945	0.12114	0.28798		0.17281	0.33965	
7	P-value(b0)	0.48366	1e-05	0.97975		-0.97975	-1e-05	
8	P-value(b1)	0.1405	0.01154	0.54161		-0.54128	-0.01121	
9	P-value(b2)	0.23661	0.01093	0.89558		-0.89233	-0.00768	
10	P-value(b3)	0.33424	0.0313	0.93327		-0.88172	0.02025	
11	P-value(b4)	0.32184	0.02202	0.92065		-0.86937	0.02926	
12	P-value(b5)	9e-04	0	0.00862	***	-0.00862	0	***
13	sd	0.23619	0.10662	0.56485		-0.20197	0.25626	
14	R^2	0.22521	0.15766	0.29394		0.60418	0.74047	
15	AIC	-5363.60113	-5490.63238	-5236.82115		-265.86525	-12.05403	
16	BIC	-5333.24343	-5460.27467	-5206.46345		-235.50755	18.30368	
17	RMSE	0.96712	0.40229	2.40325		-1.51382	0.48714	

Algorithm_2(x, y, lnz31, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.01615	-0.37656	0.33041	X	-1.96968	-1.26271	!
2	ln(Advertising.Expense)	0.2007	0.13372	0.2862	!	0.11518	0.26766	!
3	ln(Advertising.Expense)^2	0.06711	0.04068	0.09425	!	0.04419	0.09776	!
4	ln(Advertising.Expense)^3	-0.00674	-0.00993	-0.00373	!	-0.01024	-0.00404	!
5	ln(Communications.Expense)	0.07011	0.01524	0.12885	!	0.01045	0.12406	!
6	sigma	0.58345	0.51854	0.65974		0.51536	0.65656	
7	P-value(b0)	0.48782	0.00896	0.9658		-0.85506	0.10179	
8	P-value(b1)	1e-05	0	4e-05	***	0.0027	0.00274	***
9	P-value(b2)	0.00058	0	0.00427	***	-0.00082	0.00345	***
10	P-value(b3)	0.00241	0	0.02052	***	-0.01268	0.00784	***
11	P-value(b4)	0.08689	4e-05	0.58865	*	-0.38455	0.20407	*
12	P-value(b5)	0	0	0	***	0	0	***
13	sd	0.48822	0.40772	0.59265		0.5844	0.76933	
14	R^2	0.53427	0.48252	0.58489		0.18225	0.28463	
15	AIC	-446.77621	-558.24394	-331.67511		-2333.07029	-2106.50146	
16	BIC	-416.31629	-527.78401	-301.21518		-2303.1918	-2076.62297	
17	RMSE	1.07399	0.81928	1.40693		2.05616	2.64381	

Algorithm_1(x, y, lnz13, RTS="CRS", ORIENTATION="out", L=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	-0.02005	-1.26927	0.52934	X	0.4558	2.25441	!
2	ln(Advertising.Expense)	0.10566	0.04268	0.22711	!	-0.01712	0.1673	X
3	ln(Communications.Expense)	0.03728	-0.10205	0.20697	X	-0.13226	0.17676	X
4	ln(Communications.Expense)^2	0.01592	-0.02899	0.06787	X	-0.03797	0.05889	X
5	ln(Communications.Expense)^3	-0.00198	-0.00656	0.00166	X	-0.00536	0.00286	X
6	sigma	0.22663	0.16471	0.33798		0.16057	0.33384	
7	P-value(b0)	0.48826	0.00016	0.97448		-0.97447	-0.00016	
8	P-value(b1)	0.02249	0.00027	0.12161	**	-0.12161	-0.00027	**
9	P-value(b2)	0.48236	0.0542	0.96949		-0.12036	0.79494	
10	P-value(b3)	0.44834	0.02211	0.97029		-0.27992	0.66826	
11	P-value(b4)	0.39144	0.0157	0.95742		-0.62104	0.32068	
12	P-value(b5)	0.00016	0	0.00117	***	-0.00117	0	***
13	sd	0.27293	0.1461	0.56055		-0.10228	0.31216	
14	R^2	0.20499	0.14941	0.2633		0.52281	0.63669	
15	AIC	-4344.2664	-4470.24303	-4219.19144		-1273.04904	-1021.99745	
16	BIC	-4313.90869	-4439.88532	-4188.83374		-1242.69133	-991.63975	
17	RMSE	0.97188	0.47429	2.09681		-1.00645	0.61607	

Algorithm_2(x, y, lnz13, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.01817	-0.49112	0.41508	X	-5.31934	-4.41315	!
2	ln(Advertising.Expense)	0.50046	0.38971	0.62672	!	0.38281	0.61981	!
3	ln(Communications.Expense)	0.23621	-0.0589	0.55702	X	-0.0811	0.53482	X
4	ln(Communications.Expense)^2	0.01482	-0.10097	0.12669	X	-0.09701	0.13065	X
5	ln(Communications.Expense)^3	-0.00462	-0.01419	0.00588	X	-0.01526	0.00482	X
6	sigma	1.10064	0.99371	1.22263		0.28098	0.5099	
7	P-value(b0)	0.48649	0.01546	0.97531		-0.93983	0.02002	
8	P-value(b1)	0	0	0	***	0.00049	0.00049	***
9	P-value(b2)	0.24646	0.00092	0.91736		-0.27313	0.64331	
10	P-value(b3)	0.49181	0.0183	0.98074		0.70058	1.66302	
11	P-value(b4)	0.38063	0.00469	0.96071		-0.09651	0.85952	
12	P-value(b5)	0	0	0	***	0	0	***
13	sd	0.92273	0.80599	1.07171		1.14529	1.41101	
14	R^2	0.43076	0.38106	0.47908		0.12623	0.22425	
15	AIC	1611.55178	1508.28781	1718.08012		-4577.59067	-4367.79835	
16	BIC	1642.0117	1538.74773	1748.54004		-4548.03564	-4338.24333	
17	RMSE	1.56253	1.25785	1.94837		4.47712	5.16764	

Algorithm_1(x, y, lnz1C, RTS="CRS", ORIENTATION="out", L=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	-0.04591	-1.11148	0.49343	X	0.43819	2.0431	!
2	ln(Advertising.Expense)	0.1619	0.09564	0.28824	!	0.02629	0.21888	!
3	sigma	0.25803	0.19091	0.3625		0.15902	0.33061	
4	P-value(b0)	0.51021	0.00058	0.97178		-0.97177	-0.00056	
5	P-value(b1)	0.00364	0	0.03074	***	-0.03074	0	***
6	P-value(b2)	4e-05	0	0.00012	***	-0.00012	0	***
7	sd	0.30085	0.17454	0.54794		-0.03442	0.33897	
8	R^2	0.18618	0.13746	0.23563		0.47984	0.57801	
9	AIC	-3926.56988	-4051.76065	-3804.95175		-1660.58194	-1413.77304	
10	BIC	-3911.39102	-4036.58179	-3789.7729		-1645.40308	-1398.59419	
11	RMSE	1.03758	0.53448	2.03233		-0.82607	0.67178	

Algorithm_2(x, y, lnz10, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	-0.00167	-0.47778	0.39437	X	-6.24867	-5.37652	I
2	ln(Advertising.Expense)	0.66971	0.58649	0.75854	I	0.58138	0.75343	I
3	sigma	1.24493	1.12712	1.37682		0.23727	0.48698	
4	P-value(b0)	0.50222	0.01776	0.96993		-0.94176	0.01041	
5	P-value(b1)	0	0	0	***	4e-05	4e-05	***
6	P-value(b2)	0	0	0	***	0	0	***
7	sd	1.04315	0.92928	1.17718		1.3168	1.5647	
8	R^2	0.40898	0.36651	0.45038		0.12307	0.20693	
9	AIC	1850.88117	1747.43272	1957.57823		-4891.24172	-4681.09621	
10	BIC	1866.11113	1762.66269	1972.80819		-4876.5745	-4666.429	
11	RMSE	1.80258	1.46878	2.20726		5.38648	6.12497	

Algorithm_1(x, y, lnz20, RTS="CRS", ORIENTATION="out", L=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=2)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	-0.08675	-1.80506	0.59296	X	0.6822	3.08022	!
2	ln(Advertising.Expense)	0.0892	0.02372	0.25301	!	-0.09058	0.13871	X
3	ln(Advertising.Expense)^2	0.00885	0.00174	0.01986	!	-0.00241	0.0157	X
4	sigma	0.1812	0.119	0.30135		0.15458	0.33693	
5	P-value(b0)	0.50505	1e-05	0.96882		-0.96882	-1e-05	
6	P-value(b1)	0.10294	0.01119	0.33679		-0.33679	-0.01119	
7	P-value(b2)	0.08898	0.00026	0.60409	*	-0.60405	-0.00022	*
8	P-value(b3)	0.00267	0	0.03342	***	-0.03342	0	***
9	sd	0.24117	0.10271	0.59499		-0.23583	0.25646	
10	R^2	0.21363	0.14653	0.28648		0.5943	0.73425	
11	AIC	-5495.39133	-5629.19986	-5368.57457		-120.72141	139.90387	
12	BIC	-5475.15286	-5608.96139	-5348.3361		-100.48294	160.14234	
13	RMSE	1.04159	0.40643	2.649		-1.78499	0.45757	

Algorithm_2(x, y, lnz2C, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.00123	-0.457	0.33042	X	-1.45944	-0.67201	!
2	ln(Advertising.Expense)	0.29371	0.20591	0.40739	!	0.1799	0.38138	!
3	ln(Advertising.Expense)^2	0.01244	0.00113	0.02252	!	0.00253	0.02392	!
4	sigma	0.5739	0.50147	0.65933		0.48399	0.64185	
5	P-value(b0)	0.49005	0.0135	0.97325		-0.72056	0.23919	
6	P-value(b1)	0	0	0	***	9e-05	9e-05	***
7	P-value(b2)	0.11171	0	0.77771		-0.66531	0.11239	
8	P-value(b3)	0	0	0	***	0	0	***
9	sd	0.51923	0.40803	0.66933		0.4831	0.74439	
10	R^2	0.45532	0.39515	0.51532		0.20991	0.33008	
11	AIC	-675.96107	-790.77201	-559.28085		-2061.95127	-1830.46011	
12	BIC	-655.65445	-770.46539	-538.97423		-2042.0181	-1810.52695	
13	RMSE	1.17593	0.87051	1.59501		1.6362	2.3607	

Algorithm_1(x, y, lnz30, RTS="CRS", ORIENTATION="out", I =2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	-0.04953	-1.55903	0.59058	X	0.58802	2.73763	!
2	ln(Advertising.Expense)	0.07872	0.02229	0.20285	!	-0.05698	0.12359	X
3	ln(Advertising.Expense)^2	0.02144	0.00078	0.05761	!	-0.0147	0.04213	X
4	ln(Advertising.Expense)^3	-0.00158	-0.00525	0.00056	X	-0.00366	0.00214	X
5	sigma	0.18425	0.12365	0.29287		0.1746	0.34381	
6	P-value(b0)	0.491	2e-05	0.97927		-0.97927	-2e-05	
7	P-value(b1)	0.08569	0.00676	0.29327	*	-0.29325	-0.00674	*
8	P-value(b2)	0.18435	0.00858	0.81044		-0.80965	-0.00779	
9	P-value(b3)	0.33074	0.0317	0.9303		-0.87714	0.02147	
10	P-value(b4)	0.00101	0	0.01278	***	-0.01278	0	***
11	sd	0.24138	0.10746	0.55858		-0.18781	0.2633	
12	R^2	0.22049	0.15386	0.28824		0.59759	0.73197	
13	AIC	-5302.36915	-5430.27399	-5176.80701		-319.26702	-65.80004	
14	BIC	-5277.07106	-5404.97591	-5151.50892		-293.96893	-40.50195	
15	RMSE	0.98994	0.40724	2.37866		-1.46219	0.50922	

Algorithm_2(x, y, lnz30, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.01188	-0.38899	0.32949	X	-2.0262	-1.30772	!
2	ln(Advertising.Expense)	0.22766	0.16008	0.31239	!	0.1407	0.29301	!
3	ln(Advertising.Expense)^2	0.07659	0.04974	0.1047	!	0.05245	0.10741	!
4	ln(Advertising.Expense)^3	-0.00715	-0.01045	-0.004	!	-0.01075	-0.00431	!
5	sigma	0.6016	0.53359	0.68215		0.51043	0.65899	
6	P-value(b0)	0.49012	0.01181	0.97402		-0.86215	0.10005	
7	P-value(b1)	0	0	0	***	0.00092	0.00092	***
8	P-value(b2)	0.00012	0	0.00074	***	0.00056	0.0013	***
9	P-value(b3)	0.00179	0	0.01424	***	-0.00737	0.00687	***
10	P-value(b4)	0	0	0	***	0	0	***
11	sd	0.50342	0.42005	0.61314		0.59822	0.79131	
12	R^2	0.5269	0.47459	0.57713		0.18057	0.2831	
13	AIC	-387.10457	-499.39779	-271.409		-2384.13086	-2156.14206	
14	BIC	-361.72131	-474.01452	-246.02573		-2359.26763	-2131.27884	
15	RMSE	1.11324	0.84741	1.45769		2.11678	2.72706	

Algorithm_1(x, y, lnz01, RTS="CRS", ORIENTATION="out", L=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	-0.01956	-0.88234	0.45253	X	0.39085	1.72572	I
2	ln(Communications.Expense)	0.16473	0.10383	0.26866	I	0.05735	0.22217	I
3	sigma	0.21029	0.16056	0.28495		0.23852	0.3629	
4	P-value(b0)	0.51008	0.00141	0.97565		-0.97538	-0.00115	
5	P-value(b1)	0.00067	0	0.00551	***	-0.00551	0	***
6	P-value(b2)	0	0	0	***	0	0	***
7	sd	0.24216	0.14731	0.40894		0.01261	0.27425	
8	R^2	0.28033	0.21929	0.33958		0.43654	0.55683	
9	AIC	-4609.02438	-4733.38219	-4484.30256		-891.29788	-642.21825	
10	BIC	-4593.84553	-4718.20334	-4469.12371		-876.11903	-627.0394	
11	RMSE	0.91838	0.50547	1.67958		-0.54882	0.62529	

Algorithm_2(x, y, lnz01, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic Ci	Statistical Significance
1	C	-0.00024	-0.34267	0.31159	X	-5.96221	-5.30795	!
2	ln(Communications.Expense)	0.63259	0.56872	0.6998	!	0.56589	0.69697	!
3	sigma	0.9352	0.85031	1.0282		0.60369	0.78158	
4	P-value(b0)	0.50044	0.01452	0.97516		-0.94428	0.01636	
5	P-value(b1)	0	0	0	***	3e-05	3e-05	***
6	P-value(b2)	0	0	0	***	0	0	***
7	sd	0.70999	0.64208	0.78629		1.08755	1.23176	
8	R^2	0.53661	0.49295	0.57683		0.00612	0.08999	
9	AIC	1214.70628	1112.22465	1320.74542		-4007.51287	-3798.9921	
10	BIC	1229.93624	1127.45461	1335.97538		-3992.65934	-3784.13857	
11	RMSE	1.27107	1.03591	1.5362		5.31644	5.81672	

Algorithm_1(x, y, lnz02, RTS="CRS", ORIENTATION="out", L=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	-0.01638	-0.84821	0.45039	X	0.15348	1.45208	I
2	ln(Communications.Expense)	0.23206	0.12328	0.41077	I	0.04649	0.33397	I
3	ln(Communications.Expense)^2	-0.00777	-0.02046	0.00111	X	-0.01608	0.00549	X
4	sigma	0.24461	0.18913	0.32521		0.22637	0.36244	
5	P-value(b0)	0.49985	0.00127	0.97538		-0.88378	0.09034	
6	P-value(b1)	0.00212	2e-05	0.01236	***	-0.01236	-2e-05	***
7	P-value(b2)	0.23818	0.01009	0.89677		-0.7662	0.12048	
8	P-value(b3)	0	0	0	***	0	0	***
9	sd	0.27176	0.1666	0.45924		0.03191	0.32454	
10	R^2	0.27871	0.21869	0.34045		0.38428	0.50604	
11	AIC	-3863.20992	-3986.48637	-3740.5456		-1639.43977	-1393.499	
12	BIC	-3842.97145	-3966.2479	-3720.30713		-1619.2013	-1373.26052	
13	RMSE	0.90013	0.51466	1.6017		-0.30739	0.77965	

Algorithm_2(x, y, lnz02, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.01009	-0.91675	0.80745	X	-29.49577	-27.77157	!
2	ln(Communications.Expense)	3.28902	2.78946	3.80313	!	2.79343	3.8071	!
3	ln(Communications.Expense)^2	-0.18251	-0.2492	-0.11539	!	-0.25196	-0.11814	!
4	sigma	3.66	3.39243	3.94919		-0.97199	-0.41523	
5	P-value(b0)	0.49352	0.02201	0.97915		-0.50231	0.45482	
6	P-value(b1)	0	0	0	***	0.38449	0.38449	***
7	P-value(b2)	6e-05	0	0.00047	***	0.47705	0.47752	***
8	P-value(b3)	0	0	0	***	0.02053	0.02053	***
9	sd	2.8849	2.69265	3.09403		4.24471	4.64609	
10	R^2	0.47659	0.43326	0.51951		-0.08477	0.00148	
11	AIC	5179.43051	5085.73874	5272.38909		-8127.85467	-7941.20432	
12	BIC	5199.73712	5106.04535	5292.6957		-8108.32813	-7921.67777	
13	RMSE	3.64212	3.22174	4.1514		21.61786	22.54752	

Algorithm_1(x, y, lnz03, RTS="CRS", ORIENTATION="out", L=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	-0.02427	-1.17167	0.50861	X	0.36724	2.04752	!
2	ln(Communications.Expense)	0.04987	-0.08831	0.23305	X	-0.13251	0.18885	X
3	ln(Communications.Expense)^2	0.0524	0.01111	0.11293	!	-0.01135	0.09047	X
4	ln(Communications.Expense)^3	-0.00512	-0.01077	-0.00155	!	-0.00833	0.00089	X
5	sigma	0.21007	0.15344	0.30292		0.22282	0.3723	
6	P-value(b0)	0.49135	0.00023	0.97077		-0.96995	0.00059	
7	P-value(b1)	0.46052	0.05282	0.96547		-0.27664	0.63601	
8	P-value(b2)	0.09694	0.00107	0.58119	*	-0.57661	0.00351	*
9	P-value(b3)	0.05705	0.00069	0.309	*	-0.30778	0.00052	*
10	P-value(b4)	7e-05	0	0.00015	***	-0.00015	0	***
11	sd	0.25047	0.13544	0.48719		-0.06436	0.28739	
12	R^2	0.27104	0.2058	0.34111		0.46864	0.60394	
13	AIC	-4652.65792	-4780.80843	-4526.95294		-874.00724	-620.15174	
14	BIC	-4627.35983	-4755.51035	-4501.65485		-848.70915	-594.85365	
15	RMSE	0.94994	0.47618	1.93626		-0.78415	0.67593	

Algorithm_2(x, y, lnz03, RTS="CRS", ORIENTATION="out", L1=500, L2=2000, ci="all", alpha=0.05, extras= c("sigma", "P-value", "R^2", "AIC", "BIC", "RMSE"), seed=1)

	Variable	Coefficient	Lower Percentile CI	Upper Percentile CI	Statistical Significance	Lower Basic CI	Upper Basic CI	Statistical Significance
1	C	0.01505	-0.44469	0.40585	X	-7.01038	-6.15983	!
2	ln(Communications.Expense)	0.37014	0.0716	0.70936	!	0.03767	0.67544	!
3	ln(Communications.Expense)^2	0.19014	0.08395	0.29593	!	0.08583	0.29781	!
4	ln(Communications.Expense)^3	-0.0201	-0.02919	-0.01082	!	-0.02963	-0.01126	!
5	sigma	1.10622	1.00251	1.2251		0.48787	0.71045	
6	P-value(b0)	0.49154	0.01762	0.97154		-0.90687	0.04706	
7	P-value(b1)	0.09503	1e-04	0.61615	*	-0.10098	0.51508	*
8	P-value(b2)	0.01262	0	0.12648	**	-0.03895	0.08753	**
9	P-value(b3)	0.0029	0	0.02284	***	0.01065	0.03349	***
10	P-value(b4)	0	0	0	***	0	0	***
11	sd	0.82855	0.73881	0.94077		1.28519	1.48715	
12	R^2	0.54305	0.49477	0.58952		0.01349	0.10825	
13	AIC	1697.7132	1594.90135	1802.63765		-4521.3535	-4313.6172	
14	BIC	1723.09647	1620.28462	1828.02092		-4496.57964	-4288.84335	
15	RMSE	1.43719	1.16222	1.76708		5.99469	6.59955	

```
1   data <- read.csv("banks_csv.csv", header=TRUE, sep=";")
2   data <- data[ ,1:18]
3   for (i in 1:nrow(data)) {
4           if (data[i,ncol(data)]=="") data[i, ncol(data)] <- NA
5   }
6   data1 <- complete.cases(data[ ,7], data[ ,8], data[ ,9], data[ ,10], data[ ,11], data[ ,12], data[ ,13], data[ ,18])
7   data <- data[data1, ]
8   years <- data[ ,4]
9   firms <- data[ ,5]
10  unique_firms <- unique(firms)
11  all_firms <- data.frame()
12  for (i in 1:length(unique_firms)){
13          a <- which(firms==unique_firms[i])
14          b <- years[a]
15          all_firms[i,1] <- unique_firms[i]
16          all_firms[i,2] <- b[1]
17          all_firms[i,3] <- b[length(b)]
18  }
19
20  Total <- data.frame()
21  w <-1
22
23  for(lower in 1999:2013){
24          for(upper in 2000:2014){
25                  if(lower<upper){
26
27                          full_data <- data.frame()
```

```
range <- upper-lower +1
bounds <- all_firms[which(all_firms[ ,2]<=lower & all_firms[ ,3]>=upper), ]

for(j in 1:nrow(bounds)){
        from <- range*(j-1)+1
        to <- range*j
        a1 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),1]
        a2 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),2]
        a3 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),3]
        a4 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),4]
        a5 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),5]
        a6 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),6]
        a7 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),7]
        a8 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),8]
        a9 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),9]
        a10 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),10]
        a11 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),11]
        a12 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),12]
        a13 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),13]
        a14 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),14]
        a15 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),15]
        a16 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),16]
        a17 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),17]
        a18 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),18]
        if(length(a1)==range){
                full_data[from:to,1] <- a1
                full_data[from:to,2] <- a2
```

```
55                      full_data[from:to,3] <- a3
56                      full_data[from:to,4] <- a4
57                      full_data[from:to,5] <- a5
58                      full_data[from:to,6] <- a6
59                      full_data[from:to,7] <- a7
60                      full_data[from:to,8] <- a8
61                      full_data[from:to,9] <- a9
62                      full_data[from:to,10] <- a10
63                      full_data[from:to,11] <- a11
64                      full_data[from:to,12] <- a12
65                      full_data[from:to,13] <- a13
66                      full_data[from:to,14] <- a14
67                      full_data[from:to,15] <- a15
68                      full_data[from:to,16] <- a16
69                      full_data[from:to,17] <- a17
70                      full_data[from:to,18] <- a18
71              }
72      }
73
74      ok <- complete.cases(full_data)
75      full_data <- full_data[ok, ]
76      colnames(full_data) <- names(data)
77
78      total_banks <-length(unique(full_data[ ,5]))
79      total_info <- total_banks*range
80      Total[w,1] <- lower
81      Total[w,2] <- upper
```

```
82                          Total[w,3] <- total_banks
83                          Total[w,4] <- total_info
84                          w <- w+1
85                      }
86              }
87      }
88      Total <- Total[order(decreasing=TRUE, Total[ ,4]), ]
89      colnames(Total) <- c("From", "To", "Firms", "Observations")
90      row.names(Total) <- NULL
91      print(Total)
92      View(Total)
```

```
data <- read.csv("banks_csv.csv", header=TRUE, sep=";")
data <- data[ ,1:18]
for (i in 1:nrow(data)) {
        if (data[i,ncol(data)]=="") data[i, ncol(data)] <- NA
}
data1 <- complete.cases(data[ ,7], data[ ,8], data[ ,9], data[ ,10], data[ ,11], data[ ,12], data[ ,13], data[ ,18])
data <- data[data1, ]
years <- data[ ,4]
firms <- data[ ,5]
unique_firms <- unique(firms)
all_firms <- data.frame()
for (i in 1:length(unique_firms)){
        a <- which(firms==unique_firms[i])
        b <- years[a]
        all_firms[i,1] <- unique_firms[i]
        all_firms[i,2] <- b[1]
        all_firms[i,3] <- b[length(b)]
}

Total <- data.frame()
full_data <- data.frame()

## Specify date range
lower <- 2006
upper <- 2013
bounds <- all_firms[which(all_firms[ ,2]<=lower & all_firms[ ,3]>=upper), ]
range <- upper-lower +1
```

```
28
29  for(j in 1:nrow(bounds)){
30          from <- range*(j-1)+1
31          to <- range*j
32          a1 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),1]
33          a2 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),2]
34          a3 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),3]
35          a4 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),4]
36          a5 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),5]
37          a6 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),6]
38          a7 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),7]
39          a8 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),8]
40          a9 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),9]
41          a10 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),10]
42          a11 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),11]
43          a12 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),12]
44          a13 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),13]
45          a14 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),14]
46          a15 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),15]
47          a16 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),16]
48          a17 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),17]
49          a18 <- data[which(firms==bounds[j,1] & years>=lower & years<=upper),18]
50          if(length(a1)==range){
51                  full_data[from:to,1] <- a1
52                  full_data[from:to,2] <- a2
53                  full_data[from:to,3] <- a3
54                  full_data[from:to,4] <- a4
```

```
55                  full_data[from:to,5] <- a5
56                  full_data[from:to,6] <- a6
57                  full_data[from:to,7] <- a7
58                  full_data[from:to,8] <- a8
59                  full_data[from:to,9] <- a9
60                  full_data[from:to,10] <- a10
61                  full_data[from:to,11] <- a11
62                  full_data[from:to,12] <- a12
63                  full_data[from:to,13] <- a13
64                  full_data[from:to,14] <- a14
65                  full_data[from:to,15] <- a15
66                  full_data[from:to,16] <- a16
67                  full_data[from:to,17] <- a17
68                  full_data[from:to,18] <- a18
69          }
70  }
71
72  ok <- complete.cases(full_data)
73  full_data <- full_data[ok, ]
74  colnames(full_data) <- names(data)
75  path <- getwd()
76  write.table(full_data, file=paste0(path,"/balanced_panel_data.txt"), sep=" ", row.names=TRUE)
```

```
data <- read.table("balanced_panel_data.txt")
x <- data[ ,7:9]
y <- data[ ,10:11]
z <- data[ ,12:13]

new_z <- cbind(z[ ,1], z[ ,2]+1)

lnz <- log(new_z)
colnames(lnz) <- paste0("ln(",colnames(z),")")

lnz33 <- cbind(lnz[ ,1], lnz[ ,1]^2, lnz[ ,1]^3, lnz[ ,2], lnz [ ,2]^2, lnz[ ,2]^3)
colnames(lnz33) <- paste0("ln(",c(rep(colnames(z)[1],3),rep(colnames(z)[2],3)),c(")",")^2",")^3",")",")^2",")^3"))

lnz32 <- cbind(lnz[ ,1], lnz[ ,1]^2, lnz[ ,1]^3, lnz[ ,2], lnz[ ,2]^2)
colnames(lnz32) <- paste0("ln(",c(rep(colnames(z)[1],3),rep(colnames(z)[2],2)),c(")",")^2",")^3",")",")^2"))

lnz23 <- cbind(lnz[ ,1], lnz[ ,1]^2, lnz[ ,2], lnz[ ,2]^2, lnz[ ,2]^3)
colnames(lnz23) <- paste0("ln(",c(rep(colnames(z)[1],2),rep(colnames(z)[2],3)),c(")",")^2",")",")^2",")^3"))

lnz22 <- cbind(lnz[ ,1], lnz[ ,1]^2, lnz[ ,2], lnz[ ,2]^2)
colnames(lnz22) <- paste0("ln(",c(rep(colnames(z)[1],2),rep(colnames(z)[2],2)),c(")",")^2",")",")^2"))

lnz21 <- cbind(lnz[ ,1], lnz[ ,1]^2, lnz[ ,2])
colnames(lnz21) <- paste0("ln(",c(rep(colnames(z)[1],2), colnames(z)[2]),c(")",")^2",")"))

lnz12 <- cbind(lnz[ ,1], lnz[ ,2], lnz[ ,2]^2)
colnames(lnz12) <- paste0("ln(",c(colnames(z),colnames(z)[2]),c(")",")",")^2"))
```

```
28
29  lnz31 <- cbind(lnz[ ,1], lnz[ ,1]^2, lnz[ ,1]^3, lnz[ ,2])
30  colnames(lnz31) <- paste0("ln(",c(rep(colnames(z)[1],3), colnames(z)[2]),c(")",")^2",")^3",")"))
31
32  lnz13 <- cbind(lnz[ ,1], lnz[ ,2], lnz[ ,2]^2, lnz[ ,2]^3)
33  colnames(lnz13) <- paste0("ln(",c(colnames(z)[1], rep(colnames(z)[2],3)),c(")",")",")^2",")^3"))
34
35  lnz30 <- cbind(lnz[ ,1], lnz[ ,1]^2, lnz[ ,1]^3)
36  colnames(lnz30) <- paste0("ln(",rep(colnames(z)[1],3),c(")",")^2",")^3"))
37
38  lnz20 <- cbind(lnz[ ,1], lnz[ ,1]^2)
39  colnames(lnz20) <- paste0("ln(",rep(colnames(z)[1],2),c(")",")^2"))
40
41  lnz10 <- cbind(lnz[ ,1])
42  colnames(lnz10) <- paste0("ln(",colnames(z)[1],")")
43
44  lnz03 <- cbind(lnz[ ,2], lnz[ ,2]^2, lnz[ ,2]^3)
45  colnames(lnz03) <- paste0("ln(",rep(colnames(z)[2],3),c(")",")^2",")^3"))
46
47  lnz02 <- cbind(lnz[ ,2], lnz[ ,2]^2)
48  colnames(lnz02) <- paste0("ln(",rep(colnames(z)[2],2),c(")",")^2"))
49
50  lnz01 <- cbind(lnz[ ,2])
51  colnames(lnz01) <- paste0("ln(",colnames(z)[2],")")
```

```
Algorithm_1_simplified <- function(x, y, z, RTS="CRS", ORIENTATION="out", L=500, ci="percentile", alpha=0.05, seed=1) {

        if ("Benchmarking" %in% rownames(installed.packages()) == FALSE){
                install.packages("Benchmarking")
        }
        library(Benchmarking)

        if ("truncreg" %in% rownames(installed.packages()) == FALSE){
                install.packages("truncreg")
        }
        library(truncreg)

        set.seed(seed)

        if (class(x)!="matrix") x <- as.matrix(x)
        if (class(y)!="matrix") y <- as.matrix(y)
        if (class(z)!="matrix") z <- as.matrix(z)

         while (alpha<=0 || alpha>=1) {
                alpha <- readline("Choose a relatively small significance level between 0 and 1. e.g.(0.05) ")
        }
        alpha <- as.numeric(alpha)

        ## [1] DEA
        dea <- dea(x, y, RTS = RTS, ORIENTATION = ORIENTATION)
        d_hat <- eff(dea)

```

```
## [2] Truncated Regression
r <- ncol(z)
auxiliary <- c()
beta_hat <- c()

if (ORIENTATION=="out") {
        dt <- ifelse(d_hat > 1, d_hat, NA)
        model_d_hat <- truncreg(dt ~ z, point = 1, direction = "left")
}

if (ORIENTATION=="in") {
        dt <- ifelse(d_hat < 1, d_hat, NA)
        model_d_hat <- truncreg(dt ~ z, point = 1, direction = "right")
}

bad <- is.na(dt)
non_na_dt <- dt[!bad]
trunc_resid <- non_na_dt - model_d_hat$fitted.values
sd_hat <- sd(trunc_resid)
beta_hat <- summary(model_d_hat)$coefficients[1:(r+1),1]

## [3] Bootstrap Loop
bootstrap_beta = matrix(nrow=L, ncol=r+1)
bootstrap_sd = c()
m <- length(non_na_dt)
c <- c()
zb <- c()
```

```
d_star <- c()
beta_hat_star <- c()
z_m_dt <- matrix(z[complete.cases(dt,z)], ,r)

for ( i in 1:m) {
        for (j in 1:r) {
                auxiliary[j] <- z_m_dt[i,j]*beta_hat[j+1]
        }
        c[i] <- 1 - sum(auxiliary)
        zb[i] <- sum(auxiliary)
}

cc <- c/sd_hat

for ( j in 1:L) {
        ## [3.1] Drows from left-truncated normal distribution
        v <- runif(m, min=0, max=1)
        vv <- pnorm(cc) + (1 - pnorm(cc))*v
        u <- sd_hat*qnorm(vv)

        ## [3.2] computation of d*
        d_star <- zb + u

        ## [3.3] estimation of b^* and σ^*
        if (ORIENTATION=="out") {
                d_star_t <- ifelse(d_star > 1, d_star, NA)
                trunc_d_star <- truncreg(d_star_t ~ z_m_dt, point = 1, direction = "left")
```

```
82              }
83
84              if (ORIENTATION=="in") {
85                      d_star_t <- ifelse(d_star < 1, d_star, NA)
86                      trunc_d_star <- truncreg(d_star_t ~ z_m_dt, point = 1, direction = "right")
87              }
88
89              bad <- is.na(d_star_t)
90              non_na_d_star_t <- d_star_t[!bad]
91              trunc_resid_2 <- non_na_d_star_t - trunc_d_star$fitted.values
92              sd_hat_star <- sd(trunc_resid_2)
93              beta_hat_star <- summary(trunc_d_star)$coefficients[1:(r+1),1]
94
95              for (i in 1:(r+1)) {
96                      bootstrap_beta[j,i] <- beta_hat_star[i]
97              }
98              bootstrap_sd[j] <- sd_hat_star
99      }
100     A <- cbind(bootstrap_beta, bootstrap_sd)
101
102     ## [4] Confidence intervals
103     quantiles <- matrix(nrow=ncol(A), ncol=2)
104
105     for (i in 1:ncol(A)) {
106             for (j in 1:2) {
107                     quantiles[i,j] <- quantile(A[ ,i],c(alpha/2,1-(alpha/2)))[j]
108             }
```

```
109         }
110         b_ci <- matrix(nrow=ncol(A), ncol=2)
111         p_ci <- matrix(nrow=ncol(A), ncol=2)
112         statistic <- c()
113
114         ## Basic Confidence Intervals
115         if (ci=="basic") {
116                 for (i in 1:(r+1)) {
117                         b_ci[i,1] <- 2*beta_hat[i] - quantiles[i,2]
118                         b_ci[i,2] <- 2*beta_hat[i] - quantiles[i,1]
119                 }
120                 b_ci[r+2,1] <- 2*sd_hat - quantiles[r+2,2]
121                 b_ci[r+2,2] <- 2*sd_hat - quantiles[r+2,1]
122
123                 b_ci <- as.data.frame(b_ci)
124                 for (i in 1:(r+1)) {
125                         statistic <- paste0("b",i-1)
126                         b_ci <- cbind(b_ci[ ,1:2],statistic)
127                         cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[i,1], b_ci[i,2]), sep="", "\n")
128                 }
129                 statistic <- c("sd")
130                 b_ci <- cbind(b_ci[ ,1:2],statistic)
131                 cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+2,1], b_ci[r+2,2]), sep="", "\n")
132         }
133
134         ## Percentile Confidence Intervals
135         if (ci=="percentile") {
```

```
136                 for (i in 1:(r+1)) {
137                         p_ci[i,1] <- quantiles[i,1]
138                         p_ci[i,2] <- quantiles[i,2]
139                 }
140                 p_ci[r+2,1] <- quantiles[r+2,1]
141                 p_ci[r+2,2] <- quantiles[r+2,2]
142
143                 p_ci <- as.data.frame(p_ci)
144                 for (i in 1:(r+1)) {
145                         statistic <- paste0("b",i-1)
146                         p_ci <- cbind(p_ci[ ,1:2],statistic)
147                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[i,1], p_ci[i,2]), sep="", "\n")
148                 }
149                 statistic <- c("sd")
150                 p_ci <- cbind(p_ci[ ,1:2],statistic)
151                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[nrow(p_ci),1], p_ci[nrow(p_ci),2]), sep="", "\n")
152         }
153 }
```

```
1   Algorithm_2_simplified <- function(x, y, z, RTS="CRS", ORIENTATION="out", L1=100, L2=500, ci="percentile", alpha=0.05, seed=1) {
2
3           if ("Benchmarking" %in% rownames(installed.packages()) == FALSE){
4                   install.packages("Benchmarking")
5           }
6           library(Benchmarking)
7
8           if ("truncreg" %in% rownames(installed.packages()) == FALSE){
9                   install.packages("truncreg")
10          }
11          library(truncreg)
12
13          set.seed(seed)
14
15          if (class(x)!="matrix") x <- as.matrix(x)
16          if (class(y)!="matrix") y <- as.matrix(y)
17          if (class(z)!="matrix") z <- as.matrix(z)
18
19          while (alpha<=0 | | alpha>=1) {
20                  alpha <- readline("Choose a relatively small significance level between 0 and 1. e.g.(0.05) ")
21          }
22          alpha <- as.numeric(alpha)
23
24          ## [1] DEA, computation of d^
25          dea <- dea(x, y, RTS = RTS, ORIENTATION = ORIENTATION)
26          d_hat <- eff(dea)
27
```

```
## [2] Truncated Regression
r <- ncol(z)
auxiliary <- c()
beta_hat <- c()

if (ORIENTATION=="out") {
        d_hat_t <- ifelse(d_hat > 1, d_hat, NA)
        model_d_hat <- truncreg(d_hat_t ~ z, point = 1, direction = "left")
}

if (ORIENTATION=="in") {
        d_hat_t <- ifelse(d_hat < 1, d_hat, NA)
        model_d_hat <- truncreg(d_hat_t ~ z, point = 1, direction = "right")
}

bad <- is.na(d_hat_t)
non_na_d_hat_t <- d_hat_t[!bad]
trunc_resid <- non_na_d_hat_t - model_d_hat$fitted.values
sd_hat <- sd(trunc_resid)
beta_hat <- summary(model_d_hat)$coefficients[2:(r+1),1]

## [3] Bootstrap Loop 1
c <- c()
zb <- c()
n <- nrow(z)
d_star <- c()

```

```
55          ## [3.1] Drows from left-truncated normal distribution
56          for ( i in 1:n) {
57                  for (j in 1:r) {
58                          auxiliary[j] <- z[i,j]*beta_hat[j]
59                  }
60                  c[i] <- 1 - sum(auxiliary)
61                  zb[i] <- sum(auxiliary)
62          }
63
64          cc <- c/sd_hat
65          B <- matrix(nrow=n, ncol=L1)
66          for (j in 1:L1) {
67
68                  v <- runif(n, min=0, max=1)
69                  vv <- pnorm(cc) + (1 - pnorm(cc))*v
70                  u <- sd_hat*qnorm(vv)
71
72                  ## [3.2] computation of d*
73                  d_star <- zb + u
74
75                  ## [3.3] setting x* and y*
76                  x_star <- x
77                  y_star <- y*d_hat/d_star
78
79                  ## [3.4] DEA, computation of d^*
80                  dea_2 <- dea(x_star, y_star, RTS = RTS, ORIENTATION = ORIENTATION)
81                  d_hat_star <- eff(dea_2)
```

```
82
83              for (i in 1:n) {
84                      B[i,j] <- d_hat_star[i]
85              }
86      }
87      ## [4] computation of bias-corrected d^^
88      mean_di_hat_star <- c()
89
90      for (i in 1:nrow(B)) {
91              mean_di_hat_star[i] <- mean(B[i, ])
92      }
93
94      bias <- mean_di_hat_star - d_hat
95      d_2hat <- d_hat - bias
96
97      ## [5] estimation of b^^ and σ^^
98      beta_2hat <- c()
99
100     if (ORIENTATION=="out") {
101             d_2hat_t <- ifelse(d_2hat > 1, d_2hat, NA)
102             model_d_2hat <- truncreg(d_2hat_t ~ z, point = 1, direction = "left")
103     }
104
105     if (ORIENTATION=="in") {
106             d_2hat_t <- ifelse(d_2hat < 1, d_2hat, NA)
107             model_d_2hat <- truncreg(d_2hat_t ~ z, point = 1, direction = "right")
108     }
```

```
109
110         bad <- is.na(d_2hat_t)
111         non_na_d_2hat_t <- d_2hat_t[!bad]
112         trunc_resid <- non_na_d_2hat_t - model_d_2hat$fitted.values
113         sd_2hat <- sd(trunc_resid)
114         beta_2hat <- summary(model_d_2hat)$coefficients[1:(r+1),1]
115
116         ## [6] Bootstrap loop 2
117         bootstrap_beta = matrix(nrow=L2, ncol=r+1)
118         bootstrap_sd = c()
119         c_2 <- c()
120         zb_2 <- c()
121         d_2star <- c()
122         beta_2hat_star <- c()
123
124         for ( i in 1:n) {
125                 for (j in 1:r) {
126                         auxiliary[j] <- z[i,j]*beta_2hat[j+1]
127                 }
128                 c_2[i] <- 1 - sum(auxiliary)
129                 zb_2[i] <- sum(auxiliary)
130         }
131
132         cc_2 <- c_2/sd_2hat
133
134         for (j in 1:L2) {
135                 ## [6.1] Drows from left-truncated normal distribution
```

```
v_2 <- runif(n, min=0, max=1)
vv_2 <- pnorm(cc_2) + (1 - pnorm(cc_2))*v_2
u_2 <- sd_2hat*qnorm(vv_2)

## [6.2] computation of d**
d_2star <- zb_2 + u_2

## [6.3] estimation of b^^* and σ^^*
if (ORIENTATION=="out") {
        d_2star_t <- ifelse(d_2star > 1, d_2star, NA)
        model_d_2star <- truncreg(d_2star_t ~ z, point = 1, direction = "left")
}

if (ORIENTATION=="in") {
        d_2star_t <- ifelse(d_2star < 1, d_2star, NA)
        model_d_2star <- truncreg(d_2star_t ~ z, point = 1, direction = "right")
}

bad <- is.na(d_2star_t)
non_na_d_2star_t <- d_2star_t[!bad]
trunc_resid <- non_na_d_2star_t - model_d_2star$fitted.values
sd_2hat_star <- sd(trunc_resid)
beta_2hat_star <- summary(model_d_2star)$coefficients[1:(r+1),1]

for (i in 1:(r+1)) {
        bootstrap_beta[j,i] <- beta_2hat_star[i]
}
```

```
163             bootstrap_sd[j] <- sd_2hat_star
164     }
165     C <- cbind(bootstrap_beta, bootstrap_sd)
166
167     ## [7] Confidence intervals
168     quantiles <- matrix(nrow=ncol(C), ncol=2)
169
170     for (i in 1:ncol(C)) {
171             for (j in 1:2) {
172                     quantiles[i,j] <- quantile(C[ ,i],c(alpha/2,1-(alpha/2)))[j]
173             }
174     }
175     b_ci <- matrix(nrow=ncol(C), ncol=2)
176     p_ci <- matrix(nrow=ncol(C), ncol=2)
177     statistic <- c()
178
179     ## Basic Confidence Intervals
180     if (ci=="basic") {
181             for (i in 1:(r+1)) {
182                     b_ci[i,1] <- 2*beta_2hat[i] - quantiles[i,2]
183                     b_ci[i,2] <- 2*beta_2hat[i] - quantiles[i,1]
184             }
185             b_ci[r+2,1] <- 2*sd_2hat - quantiles[r+2,2]
186             b_ci[r+2,2] <- 2*sd_2hat - quantiles[r+2,1]
187
188             b_ci <- as.data.frame(b_ci)
189             for (i in 1:(r+1)) {
```

```
190                     statistic <- paste0("b",i-1)
191                     b_ci <- cbind(b_ci[ ,1:2],statistic)
192                     cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[i,1], b_ci[i,2]), sep="", "\n")
193             }
194             statistic <- c("sd")
195             b_ci <- cbind(b_ci[ ,1:2],statistic)
196             cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+2,1], b_ci[r+2,2]), sep="", "\n")
197     }
198
199     ## Percentile Confidence Intervals
200     if (ci=="percentile") {
201             for (i in 1:(r+1)) {
202                     p_ci[i,1] <- quantiles[i,1]
203                     p_ci[i,2] <- quantiles[i,2]
204             }
205             p_ci[r+2,1] <- quantiles[r+2,1]
206             p_ci[r+2,2] <- quantiles[r+2,2]
207
208             p_ci <- as.data.frame(p_ci)
209             for (i in 1:(r+1)) {
210                     statistic <- paste0("b",i-1)
211                     p_ci <- cbind(p_ci[ ,1:2],statistic)
212                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[i,1], p_ci[i,2]), sep="", "\n")
213             }
214             statistic <- c("sd")
215             p_ci <- cbind(p_ci[ ,1:2],statistic)
216             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[r+2,1], p_ci[r+2,2]), sep="", "\n")
```

```
217             }
218     }
```

```
1   Algorithm_1<- function(x, y, z, RTS="CRS", ORIENTATION="out", L=500, ci="percentile", alpha=0.05, extras=c(), seed=1, histogram=FALSE, bar=FALSE) {
2
3           if ("Benchmarking" %in% rownames(installed.packages()) == FALSE){
4                   install.packages("Benchmarking")
5           }
6           library(Benchmarking)
7
8           if ("truncreg" %in% rownames(installed.packages()) == FALSE){
9                   install.packages("truncreg")
10          }
11          library(truncreg)
12
13          if ("ggplot2" %in% rownames(installed.packages()) == FALSE){
14                  install.packages("ggplot2")
15          }
16          library(ggplot2)
17
18          set.seed(seed)
19
20          if (class(x)!="matrix") x <- as.matrix(x)
21          if (class(y)!="matrix") y <- as.matrix(y)
22          if (class(z)!="matrix") z <- as.matrix(z)
23
24          if (length(colnames(z))== 0) colnames(z) <- paste0("z",1:ncol(z))
25
26          while (alpha<=0 || alpha>=1) {
27                  alpha <- readline("Choose a relatively small significance level between 0 and 1. e.g.(0.05) ")
```

```
28          }
29          alpha <- as.numeric(alpha)
30
31          ## [1] DEA
32          dea <- dea(x, y, RTS = RTS, ORIENTATION = ORIENTATION)
33          d_hat <- eff(dea)
34
35          ## [2] Truncated Regression
36          r <- ncol(z)
37          auxiliary <- c()
38          beta_hat <- c()
39
40          if (ORIENTATION=="out") {
41                  dt <- ifelse(d_hat > 1, d_hat, NA)
42                  model_d_hat <- truncreg(dt ~ z, point = 1, direction = "left")
43          }
44
45          if (ORIENTATION=="in") {
46                  dt <- ifelse(d_hat < 1, d_hat, NA)
47                  model_d_hat <- truncreg(dt ~ z, point = 1, direction = "right")
48          }
49
50          bad <- is.na(dt)
51          non_na_dt <- dt[!bad]
52          trunc_resid <- non_na_dt - model_d_hat$fitted.values
53          sd_hat <- sd(trunc_resid)
54          beta_hat <- summary(model_d_hat)$coefficients[1:(r+1),1]
```

```
55  if ("sigma" %in% extras) sigma <- summary(model_d_hat)$coefficients[r+2,1]
56  if ("P-value" %in% extras & "sigma" %in% extras) Pr_z <- summary(model_d_hat)$coefficients[1:(r+2),4]
57  if ("P-value" %in% extras & !("sigma" %in% extras)) Pr_z <- summary(model_d_hat)$coefficients[1:(r+1),4]
58  if ("R^2" %in% extras) R_2 <- cor(model_d_hat$fitted.values, non_na_dt)^2
59  if ("AIC" %in% extras) AIC <- -2*model_d_hat$logLik + 2*length(model_d_hat$coefficients)
60  if ("BIC" %in% extras) BIC <- -2*model_d_hat$logLik + log(length(model_d_hat$fitted.values))*length(model_d_hat$coefficients)
61  if ("RMSE" %in% extras) RMSE <- sqrt(sum(trunc_resid^2)/length(trunc_resid))
62
63  ## [3] Bootstrap Loop
64  bootstrap_beta = matrix(nrow=L, ncol=r+1)
65  bootstrap_sd = c()
66  bootstrap_sigma = c()
67  bootstrap_R_2 = c()
68  if ("P-value" %in% extras & "sigma" %in% extras) bootstrap_pr_z = matrix(nrow=L, ncol=r+2)
69  if ("P-value" %in% extras & !("sigma" %in% extras)) bootstrap_pr_z = matrix(nrow=L, ncol=r+1)
70  if (!("P-value" %in% extras)) bootstrap_pr_z = c()
71  bootstrap_AIC = c()
72  bootstrap_BIC = c()
73  bootstrap_RMSE = c()
74  m <- length(non_na_dt)
75  c <- c()
76  zb <- c()
77  d_star <- c()
78  beta_hat_star <- c()
79  z_m  dt <- matrix(z[complete.cases(dt,z)], ,r)
80
81  for ( i in 1:m) {
```

```
82              for (j in 1:r) {
83                      auxiliary[j] <- z_m_dt[i,j]*beta_hat[j+1]
84              }
85              c[i] <- 1 - sum(auxiliary)
86              zb[i] <- sum(auxiliary)
87      }
88
89      cc <- c/sd_hat
90
91      for ( j in 1:L) {
92              ## [3.1] Drows from left-truncated normal distribution
93              v <- runif(m, min=0, max=1)
94              vv <- pnorm(cc) + (1 - pnorm(cc))*v
95              u <- sd_hat*qnorm(vv) #change
96
97              ## [3.2] computation of d*
98              d_star <- zb + u
99
100             ## [3.3] estimation of b^* and σ^*
101
102             if (ORIENTATION=="out") {
103                     d_star_t <- ifelse(d_star > 1, d_star, NA)
104                     trunc_d_star <- truncreg(d_star_t ~ z_m_dt, point = 1, direction = "left")
105             }
106
107             if (ORIENTATION=="in") {
108                     d_star_t <- ifelse(d_star < 1, d_star, NA)
```

```
109             trunc_d_star <- truncreg(d_star_t ~ z_m_dt, point = 1, direction = "right")
110         }
111
112         bad <- is.na(d_star_t)
113         non_na_d_star_t <- d_star_t[!bad]
114         trunc_resid_2 <- non_na_d_star_t - trunc_d_star$fitted.values
115         sd_hat_star <- sd(trunc_resid_2)
116         beta_hat_star <- summary(trunc_d_star)$coefficients[1:(r+1),1]
117         if ("sigma" %in% extras) sigma_star <- summary(trunc_d_star)$coefficients[r+2,1]
118         if ("P-value" %in% extras & "sigma" %in% extras) Pr_z_star <- summary(trunc_d_star)$coefficients[1:(r+2),4]
119         if ("P-value" %in% extras & !("sigma" %in% extras)) Pr_z_star <- summary(trunc_d_star)$coefficients[1:(r+1),4]
120         if ("R^2" %in% extras) R_2_star <- cor(trunc_d_star$fitted.values, non_na_d_star_t)^2
121         if ("AIC" %in% extras) AIC_star <- -2*trunc_d_star$logLik + 2*length(trunc_d_star$coefficients)
122         if ("BIC" %in% extras) BIC_star <- -2*trunc_d_star$logLik + log(length(trunc_d_star$fitted.values))*length(trunc_d_star$coefficients)
123         if ("RMSE" %in% extras) RMSE_star <- sqrt(sum(trunc_resid_2^2)/length(trunc_resid_2))
124
125         for (i in 1:(r+1)) {
126             bootstrap_beta[j,i] <- beta_hat_star[i]
127             if ("P-value" %in% extras & !("sigma" %in% extras)) bootstrap_pr_z[j,i] <- Pr_z_star[i]
128         }
129         if ("P-value" %in% extras & "sigma" %in% extras) {
130             for (i in 1:(r+2)) {
131                 if ("P-value" %in% extras) bootstrap_pr_z[j,i] <- Pr_z_star[i]
132             }
133         }
134         bootstrap_sd[j] <- sd_hat_star
135         if ("sigma" %in% extras) bootstrap_sigma[j] <- sigma_star
```

```
136             if ("R^2" %in% extras) bootstrap_R_2[j] <- R_2_star
137             if ("AIC" %in% extras) bootstrap_AIC[j] <- AIC_star
138             if ("BIC" %in% extras) bootstrap_BIC[j] <- BIC_star
139             if ("RMSE" %in% extras) bootstrap_RMSE[j] <- RMSE_star
140     }
141     A <- cbind(bootstrap_beta)
142     if ("sigma" %in% extras) A <- cbind(A, bootstrap_sigma)
143     if ("P-value" %in% extras) A <- cbind(A, bootstrap_pr_z)
144     A <- cbind(A, bootstrap_sd)
145     if ("R^2" %in% extras) A <- cbind(A, bootstrap_R_2)
146     if ("AIC" %in% extras) A <- cbind(A, bootstrap_AIC)
147     if ("BIC" %in% extras) A <- cbind(A, bootstrap_BIC)
148     if ("RMSE" %in% extras) A <- cbind(A, bootstrap_RMSE)
149
150     if (histogram==TRUE) {
151             for (i in 1:ncol(A)) {
152                     hist(A[ ,i])
153             }
154     }
155
156     ## [4] Confidence intervals
157     quantiles <- matrix(nrow=ncol(A), ncol=2)
158
159     for (i in 1:ncol(A)) {
160             for (j in 1:2) {
161                     quantiles[i,j] <- quantile(A[ ,i],c(alpha/2,1-(alpha/2)))[j]
162             }
```

```
163         }
164
165         bootstrap_beta <- as.data.frame(bootstrap_beta)
166         bootstrap_pr_z <- as.data.frame(bootstrap_pr_z)
167         coeff_beta <- round(sapply(bootstrap_beta, sum)/L,5)
168         names(coeff_beta) <- NULL
169         if ("P-value" %in% extras) {
170                 coeff_Pvalue <- round(sapply(bootstrap_pr_z, sum)/L,5)
171                 names(coeff_Pvalue) <- NULL
172         }
173
174         statistic <- c()
175         ## Percentile Confidence Intervals
176         if (ci=="percentile" || ci=="all") {
177                 p_ci <- matrix(nrow=ncol(A), ncol=2)
178                 p_ci <- as.data.frame(p_ci)
179                 for (i in 1:(r+1)) {
180                         p_ci[i,1] <- quantiles[i,1]
181                         p_ci[i,2] <- quantiles[i,2]
182                         statistic <- paste0("b",i-1)
183                         p_ci <- cbind(p_ci[ ,1:2],statistic)
184                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[i,1], p_ci[i,2]), sep="", "\n")
185                         if (bar==TRUE) print(ggplot(p_ci[i, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
186                         +ggtitle(paste("Percentile Confidence Interval for", statistic))+geom_hline(yintercept=0, color="red",linetype = "longdash"))
187                 }
188                 if ("sigma" %in% extras) {
189                         p_ci[r+2,1] <- quantiles[r+2,1]
```

```
p_ci[r+2,2] <- quantiles[r+2,2]
statistic <- c("sigma")
p_ci <- cbind(p_ci[ ,1:2],statistic)
cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[r+2,1], p_ci[r+2,2]), sep="", "\n")
if (bar==TRUE) print(ggplot(p_ci[r+2, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
+ggtitle(paste("Percentile Confidence Interval for", statistic))+geom_hline(yintercept=0, color="red",linetype = "longdash"))
        if ("P-value" %in% extras) {
                Pval_b <- c()
                sign_Pvalue <- c()
                for (i in (r+3):(2*r+4)) {
                p_ci[i,1] <- quantiles[i,1]
                p_ci[i,2] <- quantiles[i,2]
                statistic <- paste0("P-value(b",i-r-3,")")
                Pval_b[i-r-2] <- statistic
                p_ci <- cbind(p_ci[ ,1:2],statistic)
                cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[i,1], p_ci[i,2]), sep="", "\n")
                if (bar==TRUE) print(ggplot(p_ci[i, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                +ggtitle(paste("Percentile Confidence Interval for", statistic))
                +geom_hline(yintercept=0, color="red",linetype = "longdash"))
                if (coeff_Pvalue[i-r-2] >0.1) sign_Pvalue[i-r-2] <- " "
                if (coeff_Pvalue[i-r-2] <=0.1 & coeff_Pvalue[i-r-2] >0.05) sign_Pvalue[i-r-2] <- "*"
                if (coeff_Pvalue[i-r-2] <=0.05 & coeff_Pvalue[i-r-2] >0.01) sign_Pvalue[i-r-2] <- "**"
                if (coeff_Pvalue[i-r-2] <=0.01) sign_Pvalue[i-r-2] <- "***"
        }
        p_ci[2*r+5,1] <- quantiles[2*r+5,1]
        p_ci[2*r+5,2] <- quantiles[2*r+5,2]
        statistic <- c("sd")
```

```
217 p_ci <- cbind(p_ci[ ,1:2],statistic)
218 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[2*r+5,1], p_ci[2*r+5,2]), sep="", "\n")
219 if (bar==TRUE) print(ggplot(p_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
220 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
221 if ("R^2" %in% extras) {
222         p_ci[2*r+6,1] <- quantiles[2*r+6,1]
223         p_ci[2*r+6,2] <- quantiles[2*r+6,2]
224         statistic <- c("R^2")
225         p_ci <- cbind(p_ci[ ,1:2],statistic)
226         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
227                 p_ci[2*r+6,1], p_ci[2*r+6,2]), sep="", "\n")
228         if (bar==TRUE) print(ggplot(p_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
229         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
230         if ("AIC" %in% extras) {
231                 p_ci[2*r+7,1] <- quantiles[2*r+7,1]
232                 p_ci[2*r+7,2] <- quantiles[2*r+7,2]
233                 statistic <- c("AIC")
234                 p_ci <- cbind(p_ci[ ,1:2],statistic)
235                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
236                         p_ci[2*r+7,1], p_ci[2*r+7,2]), sep="", "\n")
237                 if (bar==TRUE) print(ggplot(p_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
238                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
239                 if ("BIC" %in% extras) {
240                         p_ci[2*r+8,1] <- quantiles[2*r+8,1]
241                         p_ci[2*r+8,2] <- quantiles[2*r+8,2]
242                         statistic <- c("BIC")
243                         p_ci <- cbind(p_ci[ ,1:2],statistic)
```

```
244                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
245                         p_ci[2*r+8,1], p_ci[2*r+8,2]), sep="", "\n")
246                     if (bar==TRUE) print(ggplot(p_ci[2*r+8, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
247                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
248                     if ("RMSE" %in% extras) {
249                         p_ci[2*r+9,1] <- quantiles[2*r+9,1]
250                         p_ci[2*r+9,2] <- quantiles[2*r+9,2]
251                         statistic <- c("RMSE")
252                         p_ci <- cbind(p_ci[ ,1:2],statistic)
253                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
254                             p_ci[2*r+9,1], p_ci[2*r+9,2]), sep="", "\n")
255                         if (bar==TRUE) print(ggplot(p_ci[2*r+9, ], aes(statistic, ymin = V1, ymax = V2))
256                         +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
257                     }
258                 }
259             }
260             if ( !("AIC" %in% extras)) {
261                 if ("BIC" %in% extras) {
262                     p_ci[2*r+7,1] <- quantiles[2*r+7,1]
263                     p_ci[2*r+7,2] <- quantiles[2*r+7,2]
264                     statistic <- c("BIC")
265                     p_ci <- cbind(p_ci[ ,1:2],statistic)
266                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
267                         p_ci[2*r+7,1], p_ci[2*r+7,2]), sep="", "\n")
268                     if (bar==TRUE) print(ggplot(p_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
269                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
270                     if ("RMSE" %in% extras) {
```

```
271                                     p_ci[2*r+8,1] <- quantiles[2*r+8,1]
272                                     p_ci[2*r+8,2] <- quantiles[2*r+8,2]
273                                     statistic <- c("RMSE")
274                                     p_ci <- cbind(p_ci[ ,1:2],statistic)
275                                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
276                                             p_ci[2*r+8,1], p_ci[2*r+8,2]), sep="", "\n")
277                                     if (bar==TRUE) print(ggplot(p_ci[2*r+8, ], aes(statistic, ymin = V1, ymax = V2))
278                                     +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
279                             }
280                     }
281                     if ( !("BIC" %in% extras)) {
282                             if ("RMSE" %in% extras) {
283                                     p_ci[2*r+7,1] <- quantiles[2*r+7,1]
284                                     p_ci[2*r+7,2] <- quantiles[2*r+7,2]
285                                     statistic <- c("RMSE")
286                                      p_ci <- cbind(p_ci[ ,1:2],statistic)
287                                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
288                                             p_ci[2*r+7,1], p_ci[2*r+7,2]), sep="", "\n")
289                                     if (bar==TRUE) print(ggplot(p_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))
290                                     +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
291                             }
292                     }
293             }
294     }
295     if ( !("R^2" %in% extras)) {
296             if ("AIC" %in% extras) {
297                     p_ci[2*r+6,1] <- quantiles[2*r+6,1]
```

```
298                 p_ci[2*r+6,2] <- quantiles[2*r+6,2]
299                 statistic <- c("AIC")
300                 p_ci <- cbind(p_ci[ ,1:2],statistic)
301                 cat("Percentile Confidence interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
302                         p_ci[2*r+6,1], p_ci[2*r+6,2]), sep="", "\n")
303                 if (bar==TRUE) print(ggplot(p_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
304                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
305                 if ("BIC" %in% extras) {
306                         p_ci[2*r+7,1] <- quantiles[2*r+7,1]
307                         p_ci[2*r+7,2] <- quantiles[2*r+7,2]
308                         statistic <- c("BIC")
309                         p_ci <- cbind(p_ci[ ,1:2],statistic)
310                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
311                                 p_ci[2*r+7,1], p_ci[2*r+7,2]), sep="", "\n")
312                         if (bar==TRUE) print(ggplot(p_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
313                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
314                         if ("RMSE" %in% extras) {
315                                 p_ci[2*r+8,1] <- quantiles[2*r+8,1]
316                                 p_ci[2*r+8,2] <- quantiles[2*r+8,2]
317                                 statistic <- c("RMSE")
318                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
319                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
320                                         p_ci[2*r+8,1], p_ci[2*r+8,2]), sep="", "\n")
321                                 if (bar==TRUE) print(ggplot(p_ci[2*r+8, ], aes(statistic, ymin = V1, ymax = V2))
322                                 +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
323                         }
324 }
```

```
325                 }
326                 if ( !("AIC" %in% extras)) {
327                         if ("BIC" %in% extras) {
328                                 p_ci[2*r+6,1] <- quantiles[2*r+6,1]
329                                 p_ci[2*r+6,2] <- quantiles[2*r+6,2]
330                                 statistic <- c("BIC")
331                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
332                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
333                                         p_ci[2*r+6,1], p_ci[2*r+6,2]), sep="", "\n")
334                                 if (bar==TRUE) print(ggplot(p_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
335                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
336                                 if ("RMSE" %in% extras) {
337                                         p_ci[2*r+7,1] <- quantiles[2*r+7,1]
338                                         p_ci[2*r+7,2] <- quantiles[2*r+7,2]
339                                         statistic <- c("RMSE")
340                                         p_ci <- cbind(p_ci[ ,1:2],statistic)
341                                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
342                                                 p_ci[2*r+7,1], p_ci[2*r+7,2]), sep="", "\n")
343                                         if (bar==TRUE) print(ggplot(p_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))
344                                         +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
345                                 }
346                         }
347                         if ( !("BIC" %in% extras)) {
348                                 if ("RMSE" %in% extras) {
349                                         p_ci[2*r+6,1] <- quantiles[2*r+6,1]
350                                         p_ci[2*r+6,2] <- quantiles[2*r+6,2]
351                                         statistic <- c("RMSE")
```

```
352                                                     p_ci <- cbind(p_ci[ ,1:2],statistic)
353                                                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
354                                                             p_ci[2*r+6,1], p_ci[2*r+6,2]), sep="", "\n")
355                                                     if (bar==TRUE) print(ggplot(p_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))
356                                                     +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
357                                             }
358                                     }
359                             }
360                     }
361             }
362
363             if ( !("P-value" %in% extras)) {
364                     p_ci[r+3,1] <- quantiles[r+3,1]
365                     p_ci[r+3,2] <- quantiles[r+3,2]
366                     statistic <- c("sd")
367                     p_ci <- cbind(p_ci[ ,1:2],statistic)
368                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[r+3,1], p_ci[r+3,2]), sep="", "\n")
369                     if (bar==TRUE) print(ggplot(p_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
370                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
371                     if ("R^2" %in% extras) {
372                             p_ci[r+4,1] <- quantiles[r+4,1]
373                             p_ci[r+4,2] <- quantiles[r+4,2]
374                             statistic <- c("R^2")
375                             p_ci <- cbind(p_ci[ ,1:2],statistic)
376                             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
377                                     p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
378                             if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
```

```
379                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
380                 if ("AIC" %in% extras) {
381                         p_ci[r+5,1] <- quantiles[r+5,1]
382                         p_ci[r+5,2] <- quantiles[r+5,2]
383                         statistic <- c("AIC")
384                         p_ci <- cbind(p_ci[ ,1:2],statistic)
385                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
386                                 p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
387                         if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
388                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
389                         if ("BIC" %in% extras) {
390                                 p_ci[r+6,1] <- quantiles[r+6,1]
391                                 p_ci[r+6,2] <- quantiles[r+6,2]
392                                 statistic <- c("BIC")
393                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
394                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
395                                         p_ci[r+6,1], p_ci[r+6,2]), sep="", "\n")
396                                 if (bar==TRUE) print(ggplot(p_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
397                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
398                                 if ("RMSE" %in% extras) {
399                                         p_ci[r+7,1] <- quantiles[r+7,1]
400                                         p_ci[r+7,2] <- quantiles[r+7,2]
401                                         statistic <- c("RMSE")
402                                         p_ci <- cbind(p_ci[ ,1:2],statistic)
403                                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
404                                                 p_ci[r+7,1], p_ci[r+7,2]), sep="", "\n")
405                                         if (bar==TRUE) print(ggplot(p_ci[r+7, ], aes(statistic, ymin = V1, ymax = V2))
```

```
406                                 +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
407                         }
408                 }
409         }
410         if ( !("AIC" %in% extras)) {
411                 if ("BIC" %in% extras) {
412                         p_ci[r+5,1] <- quantiles[r+5,1]
413                         p_ci[r+5,2] <- quantiles[r+5,2]
414                         statistic <- c("BIC")
415                         p_ci <- cbind(p_ci[ ,1:2],statistic)
416                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
417                                 p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
418                         if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
419                         +ggtitle(paste("Percentiie Confidence Interval for", statistic)))
420                         if ("RMSE" %in% extras) {
421                                 p_ci[r+6,1] <- quantiles[r+6,1]
422                                 p_ci[r+6,2] <- quantiles[r+6,2]
423                                 statistic <- c("RMSE")
424                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
425                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
426                                         p_ci[r+6,1], p_ci[r+6,2]), sep="", "\n")
427                                 if (bar==TRUE) print(ggplot(p_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))
428                                 +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
429                         }
430                 }
431                 if ( !("BIC" %in% extras)) {
432                         if ("RMSE" %in% extras) {
```

```
                                p_ci[r+5,1] <- quantiles[r+5,1]
                                p_ci[r+5,2] <- quantiles[r+5,2]
                                statistic <- c("RMSE")
                                p_ci <- cbind(p_ci[ ,1:2],statistic)
                                cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                                        p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
                                if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))
                                +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
                        }
                }
        }
}
if ( !("R^2" %in% extras)) {
        if ("AIC" %in% extras) {
                p_ci[r+4,1] <- quantiles[r+4,1]
                p_ci[r+4,2] <- quantiles[r+4,2]
                statistic <- c("AIC")
                p_ci <- cbind(p_ci[ ,1:2],statistic)
                cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                        p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
                if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                +ggtitle(paste("Percentile Confidence Interval for", statistic)))
                if ("BIC" %in% extras) {
                        p_ci[r+5,1] <- quantiles[r+5,1]
                        p_ci[r+5,2] <- quantiles[r+5,2]
                        statistic <- c("BIC")
                        p_ci <- cbind(p_ci[ ,1:2],statistic)
```

```
460                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
461                             p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
462                         if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
463                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
464                         if ("RMSE" %in% extras) {
465                             p_ci[r+6,1] <- quantiles[r+6,1]
466                             p_ci[r+6,2] <- quantiles[r+6,2]
467                             statistic <- c("RMSE")
468                             p_ci <- cbind(p_ci[ ,1:2],statistic)
469                             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
470                                 p_ci[r+6,1], p_ci[r+6,2]), sep="", "\n")
471                             if (bar==TRUE) print(ggplot(p_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))
472                             +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
473                         }
474                     }
475 }
476 if ( !("AIC" %in% extras)) {
477         if ("BIC" %in% extras) {
478             p_ci[r+4,1] <- quantiles[r+4,1]
479             p_ci[r+4,2] <- quantiles[r+4,2]
480             statistic <- c("BIC")
481             p_ci <- cbind(p_ci[ ,1:2],statistic)
482             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
483                 p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
484             if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
485             +ggtitle(paste("Percentile Confidence Interval for", statistic)))
486             if ("RMSE" %in% extras) {
```

```
487                                                     p_ci[r+5,1] <- quantiles[r+5,1]
488                                                     p_ci[r+5,2] <- quantiles[r+5,2]
489                                                     statistic <- c("RMSE")
490                                                     p_ci <- cbind(p_ci[ ,1:2],statistic)
491                                                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
492                                                             p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
493                                                     if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))
494                                                     +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
495                                             }
496                                     }
497                                     if ( !("BIC" %in% extras)) {
498                                             if ("RMSE" %in% extras) {
499                                                     p_ci[r+4,1] <- quantiles[r+4,1]
500                                                     p_ci[r+4,2] <- quantiles[r+4,2]
501                                                     statistic <- c("RMSE")
502                                                     p_ci <- cbind(p_ci[ ,1:2],statistic)
503                                                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
504                                                             p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
505                                                     if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))
506                                                     +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
507                                             }
508                                     }
509                             }
510                     }
511             }
512     }
513
```

```
514         if ("P-value" %in% extras & !("sigma" %in% extras)) {
515             Pval_b <- c()
516             sign_Pvalue <- c()
517             for (i in (r+2):(2*r+2)) {
518                 p_ci[i,1] <- quantiles[i,1]
519                 p_ci[i,2] <- quantiles[i,2]
520                 statistic <- paste0("P-value(b",i-r-2,")")
521                 Pval_b[i-r-1] <- statistic
522                 p_ci <- cbind(p_ci[ ,1:2],statistic)
523                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[i,1], p_ci[i,2]), sep="", "\n")
524                 if (bar==TRUE) print(ggplot(p_ci[i, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
525                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
526                 if (coeff_Pvalue[i-r-1] >0.1) sign_Pvalue[i-r-1] <- " "
527                 if (coeff_Pvalue[i-r-1] <=0.1 & coeff_Pvalue[i-r-1] >0.05) sign_Pvalue[i-r-1] <- "*"
528                 if (coeff_Pvalue[i-r-1] <=0.05 & coeff_Pvalue[i-r-1] >0.01) sign_Pvalue[i-r-1] <- "**"
529                 if (coeff_Pvalue[i-r-1] <=0.01) sign_Pvalue[i-r-1] <- "***"
530             }
531             p_ci[2*r+3,1] <- quantiles[2*r+3,1]
532             p_ci[2*r+3,2] <- quantiles[2*r+3,2]
533             statistic <- c("sd")
534             p_ci <- cbind(p_ci[ ,1:2],statistic)
535             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[2*r+3,1], p_ci[2*r+3,2]), sep="", "\n")
536             if (bar==TRUE) print(ggplot(p_ci[2*r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
537             +ggtitle(paste("Percentile Confidence Interval for", statistic)))
538             if ("R^2" %in% extras) {
539                 p_ci[2*r+4,1] <- quantiles[2*r+4,1]
540                 p_ci[2*r+4,2] <- quantiles[2*r+4,2]
```

```
statistic <- c("R^2")
p_ci <- cbind(p_ci[ ,1:2],statistic)
cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[2*r+4,1], p_ci[2*r+4,2]), sep="", "\n")
if (bar==TRUE) print(ggplot(p_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
+ggtitle(paste("Percentile Confidence Interval for", statistic)))
if ("AIC" %in% extras) {
        p_ci[2*r+5,1] <- quantiles[2*r+5,1]
        p_ci[2*r+5,2] <- quantiles[2*r+5,2]
        statistic <- c("AIC")
        p_ci <- cbind(p_ci[ ,1:2],statistic)
        cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                p_ci[2*r+5,1], p_ci[2*r+5,2]), sep="", "\n")
        if (bar==TRUE) print(ggplot(p_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
        +ggtitle(paste("Percentile Confidence Interval for", statistic)))
        if ("BIC" %in% extras) {
                p_ci[2*r+6,1] <- quantiles[2*r+6,1]
                p_ci[2*r+6,2] <- quantiles[2*r+6,2]
                statistic <- c("BIC")
                p_ci <- cbind(p_ci[ ,1:2],statistic)
                cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                        p_ci[2*r+6,1], p_ci[2*r+6,2]), sep="", "\n")
                if (bar==TRUE) print(ggplot(p_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                +ggtitle(paste("Percentile Confidence Interval for", statistic)))
                if ("RMSE" %in% extras) {
                        p_ci[2*r+7,1] <- quantiles[2*r+7,1]
                        p_ci[2*r+7,2] <- quantiles[2*r+7,2]
                        statistic <- c("RMSE")
```

```
568                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
569                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
570                                         p_ci[2*r+7,1], p_ci[2*r+7,2]), sep="", "\n")
571                                 if (bar==TRUE) print(ggplot(p_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
572                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
573                         }
574                 }
575         }
576         if ( !("AIC" %in% extras)) {
577                 if ("BIC" %in% extras) {
578                         p_ci[2*r+5,1] <- quantiles[2*r+5,1]
579                         p_ci[2*r+5,2] <- quantiles[2*r+5,2]
580                         statistic <- c("BIC")
581                         p_ci <- cbind(p_ci[ ,1:2],statistic)
582                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
583                                 p_ci[2*r+5,1], p_ci[2*r+5,2]), sep="", "\n")
584                         if (bar==TRUE) print(ggplot(p_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
585                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
586                         if ("RMSE" %in% extras) {
587                                 p_ci[2*r+6,1] <- quantiles[2*r+6,1]
588                                 p_ci[2*r+6,2] <- quantiles[2*r+6,2]
589                                 statistic <- c("RMSE")
590                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
591                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
592                                         p_ci[2*r+6,1], p_ci[2*r+6,2]), sep="", "\n")
593                                 if (bar==TRUE) print(ggplot(p_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
594                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
```

```
595                                     }
596                                 }
597                                 if ( !("BIC" %in% extras)) {
598                                     if ("RMSE" %in% extras) {
599                                         p_ci[2*r+5,1] <- quantiles[2*r+5,1]
600                                         p_ci[2*r+5,2] <- quantiles[2*r+5,2]
601                                         statistic <- c("RMSE")
602                                         p_ci <- cbind(p_ci[ ,1:2],statistic)
603                                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
604                                             p_ci[2*r+5,1], p_ci[2*r+5,2]), sep="", "\n")
605                                         if (bar==TRUE) print(ggplot(p_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
606                                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
607                                     }
608                                 }
609                             }
610                         }
611                         if ( !("R^2" %in% extras)) {
612                             if ("AIC" %in% extras) {
613                                 p_ci[2*r+4,1] <- quantiles[2*r+4,1]
614                                 p_ci[2*r+4,2] <- quantiles[2*r+4,2]
615                                 statistic <- c("AIC")
616                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
617                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
618                                     p_ci[2*r+4,1], p_ci[2*r+4,2]), sep="", "\n")
619                                 if (bar==TRUE) print(ggplot(p_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
620                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
621                                 if ("BIC" %in% extras) {
```

```
622                         p_ci[2*r+5,1] <- quantiles[2*r+5,1]
623                         p_ci[2*r+5,2] <- quantiles[2*r+5,2]
624                         statistic <- c("BIC")
625                         p_ci <- cbind(p_ci[ ,1:2],statistic)
626                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
627                                 p_ci[2*r+5,1], p_ci[2*r+5,2]), sep="", "\n")
628                         if (bar==TRUE) print(ggplot(p_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
629                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
630                         if ("RMSE" %in% extras) {
631                                 p_ci[2*r+6,1] <- quantiles[2*r+6,1]
632                                 p_ci[2*r+6,2] <- quantiles[2*r+6,2]
633                                 statistic <- c("RMSE")
634                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
635                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
636                                         p_ci[2*r+6,1], p_ci[2*r+6,2]), sep="", "\n")
637                                 if (bar==TRUE) print(ggplot(p_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
638                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
639                         }
640                 }
641         }
642         if ( !("AIC" %in% extras)) {
643                 if ("BIC" %in% extras) {
644                         p_ci[2*r+4,1] <- quantiles[2*r+4,1]
645                         p_ci[2*r+4,2] <- quantiles[2*r+4,2]
646                         statistic <- c("BIC")
647                         p_ci <- cbind(p_ci[ ,1:2],statistic)
648                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
```

```
649                         p_ci[2*r+4,1], p_ci[2*r+4,2]), sep="", "\n")
650                     if (bar==TRUE) print(ggplot(p_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
651                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
652                     if ("RMSE" %in% extras) {
653                         p_ci[2*r+5,1] <- quantiles[2*r+5,1]
654                         p_ci[2*r+5,2] <- quantiles[2*r+5,2]
655                         statistic <- c("RMSE")
656                         p_ci <- cbind(p_ci[ ,1:2],statistic)
657                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
658                             p_ci[2*r+5,1], p_ci[2*r+5,2]), sep="", "\n")
659                         if (bar==TRUE) print(ggplot(p_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
660                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
661                     }
662                 }
663                 if ( !("BIC" %in% extras)) {
664                     if ("RMSE" %in% extras) {
665                         p_ci[2*r+4,1] <- quantiles[2*r+4,1]
666                         p_ci[2*r+4,2] <- quantiles[2*r+4,2]
667                         statistic <- c("RMSE")
668                         p_ci <- cbind(p_ci[ ,1:2],statistic)
669                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
670                             p_ci[2*r+4,1], p_ci[2*r+4,2]), sep="", "\n")
671                         if (bar==TRUE) print(ggplot(p_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
672                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
673                     }
674                 }
675             }
```

```
676                 }
677         }
678
679         if ( !("P-value" %in% extras) & !("sigma" %in% extras)) {
680                 p_ci[r+2,1] <- quantiles[r+2,1]
681                 p_ci[r+2,2] <- quantiles[r+2,2]
682                 statistic <- c("sd")
683                 p_ci <- cbind(p_ci[ ,1:2],statistic)
684                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[r+2,1], p_ci[r+2,2]), sep="", "\n")
685                 if (bar==TRUE) print(ggplot(p_ci[r+2, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
686                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
687                 if ("R^2" %in% extras) {
688                         p_ci[r+3,1] <- quantiles[r+3,1]
689                         p_ci[r+3,2] <- quantiles[r+3,2]
690                         statistic <- c("R^2")
691                         p_ci <- cbind(p_ci[ ,1:2],statistic)
692                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[r+3,1], p_ci[r+3,2]), sep="", "\n")
693                         if (bar==TRUE) print(ggplot(p_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
694                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
695                         if ("AIC" %in% extras) {
696                                 p_ci[r+4,1] <- quantiles[r+4,1]
697                                 p_ci[r+4,2] <- quantiles[r+4,2]
698                                 statistic <- c("AIC")
699                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
700                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
701                                         p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
702                                 if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
```

```
703                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
704                     if ("BIC" %in% extras) {
705                             p_ci[r+5,1] <- quantiles[r+5,1]
706                             p_ci[r+5,2] <- quantiles[r+5,2]
707                             statistic <- c("BIC")
708                             p_ci <- cbind(p_ci[ ,1:2],statistic)
709                             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
710                                     p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
711                             if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
712                             +ggtitle(paste("Percentile Confidence Interval for", statistic)))
713                             if ("RMSE" %in% extras) {
714                                     p_ci[r+6,1] <- quantiles[r+6,1]
715                                     p_ci[r+6,2] <- quantiles[r+6,2]
716                                     statistic <- c("RMSE")
717                                     p_ci <- cbind(p_ci[ ,1:2],statistic)
718                                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
719                                             p_ci[r+6,1], p_ci[r+6,2]), sep="", "\n")
720                                     if (bar==TRUE) print(ggplot(p_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
721                                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
722                             }
723                     }
724             }
725             if ( !("AIC" %in% extras)) {
726                     if ("BIC" %in% extras) {
727                             p_ci[r+4,1] <- quantiles[r+4,1]
728                             p_ci[r+4,2] <- quantiles[r+4,2]
729                             statistic <- c("BIC")
```

```
730                     p_ci <- cbind(p_ci[ ,1:2],statistic)
731                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
732                             p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
733                     if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
734                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
735                     if ("RMSE" %in% extras) {
736                             p_ci[r+5,1] <- quantiles[r+5,1]
737                             p_ci[r+5,2] <- quantiles[r+5,2]
738                             statistic <- c("RMSE")
739                             p_ci <- cbind(p_ci[ ,1:2],statistic)
740                             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
741                                     p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
742                             if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
743                             +ggtitle(paste("Percentile Confidence Interval for", statistic)))
744                     }
745             }
746             if ( !("BIC" %in% extras)) {
747                     if ("RMSE" %in% extras) {
748                             p_ci[r+4,1] <- quantiles[r+4,1]
749                             p_ci[r+4,2] <- quantiles[r+4,2]
750                             statistic <- c("RMSE")
751                             p_ci <- cbind(p_ci[ ,1:2],statistic)
752                             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
753                                     p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
754                             if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
755                             +ggtitle(paste("Percentile Confidence Interval for", statistic)))
756                     }
```

```
757                         }
758                 }
759         }
760         if ( !("R^2" %in% extras)) {
761                 if ("AIC" %in% extras) {
762                         p_ci[r+3,1] <- quantiles[r+3,1]
763                         p_ci[r+3,2] <- quantiles[r+3,2]
764                         statistic <- c("AIC")
765                         p_ci <- cbind(p_ci[ ,1:2],statistic)
766                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
767                                 p_ci[r+3,1], p_ci[r+3,2]), sep="", "\n")
768                         If (bar==TRUE) print(ggplot(p_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
769                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
770                         if ("BIC" %in% extras) {
771                                 p_ci[r+4,1] <- quantiles[r+4,1]
772                                 p_ci[r+4,2] <- quantiles[r+4,2]
773                                 statistic <- c("BIC")
774                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
775                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
776                                         p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
777                                 if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
778                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
779                                 if ("RMSE" %in% extras) {
780                                         p_ci[r+5,1] <- quantiles[r+5,1]
781                                         p_ci[r+5,2] <- quantiles[r+5,2]
782                                         statistic <- c("RMSE")
783                                         p_ci <- cbind(p_ci[ ,1:2],statistic)
```

```
784                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
785                                         p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
786                                 if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
787                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
788                         }
789                 }
790         }
791         if ( !("AIC" %in% extras)) {
792                 if ("BIC" %in% extras) {
793                         p_ci[r+3,1] <- quantiles[r+3,1]
794                         p_ci[r+3,2] <- quantiles[r+3,2]
795                         statistic <- c("BIC")
796                         p_ci <- cbind(p_ci[ ,1:2],statistic)
797                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
798                                 p_ci[r+3,1], p_ci[r+3,2]), sep="", "\n")
799                         if (bar==TRUE) print(ggplot(p_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
800                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
801                         if ("RMSE" %in% extras) {
802                                 p_ci[r+4,1] <- quantiles[r+4,1]
803                                 p_ci[r+4,2] <- quantiles[r+4,2]
804                                 statistic <- c("RMSE")
805                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
806                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
807                                         p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
808                                 if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
809                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
810                         }
```

```
811                                     }
812                                     if ( !("BIC" %in% extras)) {
813                                             if ("RMSE" %in% extras) {
814                                                     p_ci[r+3,1] <- quantiles[r+3,1]
815                                                     p_ci[r+3,2] <- quantiles[r+3,2]
816                                                     statistic <- c("RMSE")
817                                                     p_ci <- cbind(p_ci[ ,1:2],statistic)
818                                                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
819                                                             p_ci[r+3,1], p_ci[r+3,2]), sep="", "\n")
820                                                     if (bar==TRUE) print(ggplot(p_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
821                                                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
822                                             }
823                                     }
824                             }
825                     }
826             }
827     }
828
829     ## Basic Confidence Intervals
830     if (ci=="basic" || ci=="all") {
831             b_ci <- matrix(nrow=ncol(A), ncol=2)
832             b_ci <- as.data.frame(b_ci)
833             for (i in 1:(r+1)) {
834                     b_ci[i,1] <- 2*beta_hat[i] - quantiles[i,2]
835                     b_ci[i,2] <- 2*beta_hat[i] - quantiles[i,1]
836                     statistic <- paste0("b",i-1)
837                     b_ci <- cbind(b_ci[ ,1:2],statistic)
```

```
838             cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[i,1], b_ci[i,2]), sep="", "\n")
839             if (bar==TRUE) print(ggplot(b_ci[i, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
840             +ggtitle(paste("Basic Confidence Interval for", statistic))+geom_hline(yintercept=0, color="red",linetype = "longdash"))
841         }
842         if ("sigma" %in% extras) {
843             b_ci[r+2,1] <- 2*sigma - quantiles[r+2,2]
844             b_ci[r+2,2] <- 2*sigma - quantiles[r+2,1]
845             statistic <- c("sigma")
846             b_ci <- cbind(b_ci[ ,1:2],statistic)
847             cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+2,1], b_ci[r+2,2]), sep="", "\n")
848             if (bar==TRUE) print(ggplot(b_ci[r+2, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
849             +ggtitle(paste("Basic Confidence Interval for", statistic)))
850             if ("P-value" %in% extras) {
851                 Pval_b <- c()
852                 sign_Pvalue <- c()
853                 for (i in (r+3):(2*r+4)) {
854                     b_ci[i,1] <- 2*Pr_z[i-r-2] - quantiles[i,2]
855                     b_ci[i,2] <- 2*Pr_z[i-r-2] - quantiles[i,1]
856                     statistic <- paste0("P-value(b",i-r-3,")")
857                     Pval_b[i-r-2] <- statistic
858                     b_ci <- cbind(b_ci[ ,1:2],statistic)
859                     cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[i,1], b_ci[i,2]), sep="", "\n")
860                     if (bar==TRUE) print(ggplot(b_ci[i, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
861                     +ggtitle(paste("Basic Confidence Interval for", statistic)))
862                     if (coeff_Pvalue[i-r-2] >0.1) sign_Pvalue[i-r-2] <- " "
863                     if (coeff_Pvalue[i-r-2] <=0.1 & coeff_Pvalue[i-r-2] >0.05) sign_Pvalue[i-r-2] <- "*"
864                     if (coeff_Pvalue[i-r-2] <=0.05 & coeff_Pvalue[i-r-2] >0.01) sign_Pvalue[i-r-2] <- "**"
```

```
865                 if (coeff_Pvalue[i-r-2] <=0.01) sign_Pvalue[i-r-2] <- "***"
866         }
867         b_ci[2*r+5,1] <- 2*sd_hat - quantiles[2*r+5,2]
868         b_ci[2*r+5,2] <- 2*sd_hat - quantiles[2*r+5,1]
869         statistic <- c("sd")
870         b_ci <- cbind(b_ci[ ,1:2],statistic)
871         cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[2*r+5,1], b_ci[2*r+5,2]), sep="", "\n")
872         if (bar==TRUE) print(ggplot(b_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
873         +ggtitle(paste("Basic Confidence Interval for", statistic)))
874         if ("R^2" %in% extras) {
875                 b_ci[2*r+6,1] <- 2*R_2 - quantiles[2*r+6,2]
876                 b_ci[2*r+6,2] <- 2*R_2 - quantiles[2*r+6,1]
877                 statistic <- c("R^2")
878                 b_ci <- cbind(b_ci[ ,1:2],statistic)
879                 cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
880                         b_ci[2*r+6,1], b_ci[2*r+6,2]), sep="", "\n")
881                 if (bar==TRUE) print(ggplot(b_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
882                 +ggtitle(paste("Basic Confidence Interval for", statistic)))
883                 if ("AIC" %in% extras) {
884                         b_ci[2*r+7,1] <- 2*AIC - quantiles[2*r+7,2]
885                         b_ci[2*r+7,2] <- 2*AIC - quantiles[2*r+7,1]
886                         statistic <- c("AIC")
887                         b_ci <- cbind(b_ci[ ,1:2],statistic)
888                         cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
889                                 b_ci[2*r+7,1], b_ci[2*r+7,2]), sep="", "\n")
890                         if (bar==TRUE) print(ggplot(b_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
891                         +ggtitle(paste("Basic Confidence Interval for", statistic)))
```

```
892                 if ("BIC" %in% extras) {
893                         b_ci[2*r+8,1] <- 2*BIC - quantiles[2*r+8,2]
894                         b_ci[2*r+8,2] <- 2*BIC - quantiles[2*r+8,1]
895                         statistic <- c("BIC")
896                         b_ci <- cbind(b_ci[ ,1:2],statistic)
897                         cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
898                                 b_ci[2*r+8,1], b_ci[2*r+8,2]), sep="", "\n")
899                         if (bar==TRUE) print(ggplot(b_ci[2*r+8, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
900                         +ggtitle(paste("Basic Confidence Interval for", statistic)))
901                         if ("RMSE" %in% extras) {
902                                 b_ci[2*r+9,1] <- 2*RMSE - quantiles[2*r+9,2]
903                                 b_ci[2*r+9,2] <- 2*RMSE - quantiles[2*r+9,1]
904                                 statistic <- c("RMSE")
905                                 b_ci <- cbind(b_ci[ ,1:2],statistic)
906                                 cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
907                                         b_ci[2*r+9,1], b_ci[2*r+9,2]), sep="", "\n")
908                                 if (bar==TRUE) print(ggplot(b_ci[2*r+9, ], aes(statistic, ymin = V1, ymax = V2))
909                                 +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
910                         }
911                 }
912         }
913         if ( !("AIC" %in% extras)) {
914                 if ("BIC" %in% extras) {
915                         b_ci[2*r+7,1] <- 2*BIC - quantiles[2*r+7,2]
916                         b_ci[2*r+7,2] <- 2*BIC - quantiles[2*r+7,1]
917                         statistic <- c("BIC")
918                         b_ci <- cbind(b_ci[ ,1:2],statistic)
```

```
919                 cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
920                         b_ci[2*r+7,1], b_ci[2*r+7,2]), sep="", "\n")
921                 if (bar==TRUE) print(ggplot(b_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
922                 +ggtitle(paste("Basic Confidence Interval for", statistic)))
923                 if ("RMSE" %in% extras) {
924                         b_ci[2*r+8,1] <- 2*RMSE - quantiles[2*r+8,2]
925                         b_ci[2*r+8,2] <- 2*RMSE - quantiles[2*r+8,1]
926                         statistic <- c("RMSE")
927                         b_ci <- cbind(b_ci[ ,1:2],statistic)
928                         cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
929                                 b_ci[2*r+8,1], b_ci[2*r+8,2]), sep="", "\n")
930                         if (bar==TRUE) print(ggplot(b_ci[2*r+8, ], aes(statistic, ymin = V1, ymax = V2))
931                         +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
932                 }
933         }
934         if ( l("BIC" %in% extras)) {
935                 if ("RMSE" %in% extras) {
936                         b_ci[2*r+7,1] <- 2*RMSE - quantiles[2*r+7,2]
937                         b_ci[2*r+7,2] <- 2*RMSE - quantiles[2*r+7,1]
938                         statistic <- c("RMSE")
939                         b_ci <- cbind(b_ci[ ,1:2],statistic)
940                         cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
941                                 b_ci[2*r+7,1], b_ci[2*r+7,2]), sep="", "\n")
942                         if (bar==TRUE) print(ggplot(b_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))
943                         +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
944                 }
945         }
```

```
946                     }
947             }
948             if ( !("R^2" %in% extras)) {
949                     if ("AIC" %in% extras) {
950                             b_ci[2*r+6,1] <- 2*AIC - quantiles[2*r+6,2]
951                             b_ci[2*r+6,2] <- 2*AIC - quantiles[2*r+6,1]
952                             statistic <- c("AIC")
953                             b_ci <- cbind(b_ci[ ,1:2],statistic)
954                             cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
955                                     b_ci[2*r+6,1], b_ci[2*r+6,2]), sep="", "\n")
956                             if (bar==TRUE) print(ggplot(b_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
957                             +ggtitle(paste("Basic Confidence Interval for", statistic)))
958                             if ("BIC" %in% extras) {
959                                     b_ci[2*r+7,1] <- 2*BIC - quantiles[2*r+7,2]
960                                     b_ci[2*r+7,2] <- 2*BIC - quantiles[2*r+7,1]
961                                     statistic <- c("BIC")
962                                     b_ci <- cbind(b_ci[ ,1:2],statistic)
963                                     cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
964                                             b_ci[2*r+7,1], b_ci[2*r+7,2]), sep="", "\n")
965                                     if (bar==TRUE) print(ggplot(b_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
966                                     +ggtitle(paste("Basic Confidence Interval for", statistic)))
967                                     if ("RMSE" %in% extras) {
968                                             b_ci[2*r+8,1] <- 2*RMSE - quantiles[2*r+8,2]
969                                             b_ci[2*r+8,2] <- 2*RMSE - quantiles[2*r+8,1]
970                                             statistic <- c("RMSE")
971                                             b_ci <- cbind(b_ci[ ,1:2],statistic)
972                                             cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
```

```
                                        b_ci[2*r+8,1], b_ci[2*r+8,2]), sep="", "\n")
                                if (bar==TRUE) print(ggplot(b_ci[2*r+8, ], aes(statistic, ymin = V1, ymax = V2))
                                +geom_errorbar()+ggtitie(paste("Basic Confidence Interval for", statistic)))
                        }
                }
        }
        if ( !("AIC" %in% extras)) {
                if ("BIC" %in% extras) {
                        b_ci[2*r+6,1] <- 2*BIC - quantiles[2*r+6,2]
                        b_ci[2*r+6,2] <- 2*BIC - quantiles[2*r+6,1]
                        statistic <- c("BIC")
                        b_ci <- cbind(b_ci[ ,1:2],statistic)
                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                                b_ci[2*r+6,1], b_ci[2*r+6,2]), sep="", "\n")
                        if (bar==TRUE) print(ggplot(b_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                        +ggtitle(paste("Basic Confidence Interval for", statistic)))
                        if ("RMSE" %in% extras) {
                                b_ci[2*r+7,1] <- 2*RMSE - quantiles[2*r+7,2]
                                b_ci[2*r+7,2] <- 2*RMSE - quantiles[2*r+7,1]
                                statistic <- c("RMSE")
                                b_ci <- cbind(b_ci[ ,1:2],statistic)
                                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                                        b_ci[2*r+7,1], b_ci[2*r+7,2]), sep="", "\n")
                                if (bar==TRUE) print(ggplot(b_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))
                                +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
                        }
                }
```

```
1000                                    if ( !("BIC" %in% extras)) {
1001                                            if ("RMSE" %in% extras) {
1002                                                    b_ci[2*r+6,1] <- 2*RMSE - quantiles[2*r+6,2]
1003                                                    b_ci[2*r+6,2] <- 2*RMSE - quantiles[2*r+6,1]
1004                                                    statistic <- c("RMSE")
1005                                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1006                                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1007                                                            b_ci[2*r+6,1], b_ci[2*r+6,2]), sep="", "\n")
1008                                                    if (bar==TRUE) print(ggplot(b_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))
1009                                                    +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1010                                            }
1011                                    }
1012                            }
1013                    }
1014            }
1015
1016            if ( !("P-value" %in% extras)) {
1017                    b_ci[r+3,1] <- 2*sd_hat - quantiles[r+3,2]
1018                    b_ci[r+3,2] <- 2*sd_hat - quantiles[r+3,1]
1019                    statistic <- c("sd")
1020                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1021                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+3,1], b_ci[r+3,2]), sep="", "\n")
1022                    if (bar==TRUE) print(ggplot(b_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1023                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1024                    if ("R^2" %in% extras) {
1025                            b_ci[r+4,1] <- 2*R_2 - quantiles[r+4,2]
1026                            b_ci[r+4,2] <- 2*R_2 - quantiles[r+4,1]
```

```
statistic <- c("R^2")
b_ci <- cbind(b_ci[ ,1:2],statistic)
cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
+ggtitle(paste("Basic Confidence Interval for", statistic)))
if ("AIC" %in% extras) {
        b_ci[r+5,1] <- 2*AIC - quantiles[r+5,2]
        b_ci[r+5,2] <- 2*AIC - quantiles[r+5,1]
        statistic <- c("AIC")
        b_ci <- cbind(b_ci[ ,1:2],statistic)
        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
        if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
        +ggtitle(paste("Basic Confidence Interval for", statistic)))
        if ("BIC" %in% extras) {
                b_ci[r+6,1] <- 2*BIC - quantiles[r+6,2]
                b_ci[r+6,2] <- 2*BIC - quantiles[r+6,1]
                statistic <- c("BIC")
                b_ci <- cbind(b_ci[ ,1:2],statistic)
                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                        b_ci[r+6,1], b_ci[r+6,2]), sep="", "\n")
                if (bar==TRUE) print(ggplot(b_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                +ggtitle(paste("Basic Confidence Interval for", statistic)))
                if ("RMSE" %in% extras) {
                        b_ci[r+7,1] <- 2*RMSE - quantiles[r+7,2]
                        b_ci[r+7,2] <- 2*RMSE - quantiles[r+7,1]
                        statistic <- c("RMSE")
```

```
1054                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1055                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1056                                            b_ci[r+7,1], b_ci[r+7,2]), sep="", "\n")
1057                                    if (bar==TRUE) print(ggplot(b_ci[r+7, ], aes(statistic, ymin = V1, ymax = V2))
1058                                    +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1059                            }
1060                    }
1061    }
1062    if ( !("AIC" %in% extras)) {
1063            if ("BIC" %in% extras) {
1064                    b_ci[r+5,1] <- 2*BIC - quantiles[r+5,2]
1065                    b_ci[r+5,2] <- 2*BIC - quantiles[r+5,1]
1066                    statistic <- c("BIC")
1067                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1068                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1069                            b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
1070                    if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1071                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1072                    if ("RMSE" %in% extras) {
1073                            b_ci[r+6,1] <- 2*RMSE - quantiles[r+6,2]
1074                            b_ci[r+6,2] <- 2*RMSE - quantiles[r+6,1]
1075                            statistic <- c("RMSE")
1076                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1077                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1078                                    b_ci[r+6,1], b_ci[r+6,2]), sep="", "\n")
1079                            If (bar==TRUE) print(ggplot(b_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))
1080                            +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
```

```
1081                                    }
1082                                }
1083                                if ( !("BIC" %in% extras)) {
1084                                    if ("RMSE" %in% extras) {
1085                                        b_ci[r+5,1] <- 2*BIC - quantiles[r+5,2]
1086                                        b_ci[r+5,2] <- 2*BIC - quantiles[r+5,1]
1087                                        statistic <- c("RMSE")
1088                                        b_ci <- cbind(b_ci[ ,1:2],statistic)
1089                                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1090                                            b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
1091                                        if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))
1092                                        +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1093                                    }
1094                                }
1095                            }
1096                        }
1097                        if ( !("R^2" %in% extras)) {
1098                            if ("AIC" %in% extras) {
1099                                b_ci[r+4,1] <- 2*AIC - quantiles[r+4,2]
1100                                b_ci[r+4,2] <- 2*AIC - quantiles[r+4,1]
1101                                statistic <- c("AIC")
1102                                b_ci <- cbind(b_ci[ ,1:2],statistic)
1103                                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1104                                    b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1105                                if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1106                                +ggtitle(paste("Basic Confidence Interval for", statistic)))
1107                                if ("BIC" %in% extras) {
```

```
1108                    b_ci[r+5,1] <- 2*BIC - quantiles[r+5,2]
1109                    b_ci[r+5,2] <- 2*BIC - quantiles[r+5,1]
1110                    statistic <- c("BIC")
1111                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1112                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1113                            b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
1114                    if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1115                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1116                    if ("RMSE" %in% extras) {
1117                            b_ci[r+6,1] <- 2*RMSE - quantiles[r+6,2]
1118                            b_ci[r+6,2] <- 2*RMSE - quantiles[r+6,1]
1119                            statistic <- c("RMSE")
1120                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1121                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1122                                    b_ci[r+6,1], b_ci[r+6,2]), sep="", "\n")
1123                            if (bar==TRUE) print(ggplot(b_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))
1124                            +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1125                    }
1126            }
1127    }
1128    if ( !("AIC" %in% extras)) {
1129            if ("BIC" %in% extras) {
1130                    b_ci[r+4,1] <- 2*BIC - quantiles[r+4,2]
1131                    b_ci[r+4,2] <- 2*BIC - quantiles[r+4,1]
1132                    statistic <- c("BIC")
1133                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1134                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
```

```
1135                                    b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1136                            if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1137                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1138                            if ("RMSE" %in% extras) {
1139                                    b_ci[r+5,1] <- 2*RMSE - quantiles[r+5,2]
1140                                    b_ci[r+5,2] <- 2*RMSE - quantiles[r+5,1]
1141                                    statistic <- c("RMSE")
1142                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1143                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1144                                            b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
1145                                    if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))
1146                                    +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1147                            }
1148                    }
1149                    if ( !("BIC" %in% extras)) {
1150                            if ("RMSE" %in% extras) {
1151                                    b_ci[r+4,1] <- 2*RMSE - quantiles[r+4,2]
1152                                    b_ci[r+4,2] <- 2*RMSE - quantiles[r+4,1]
1153                                    statistic <- c("RMSE")
1154                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1155                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1156                                            b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1157                                    if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))
1158                                    +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1159                            }
1160                    }
1161            }
```

```
1162                        }
1163                }
1164        }
1165
1166        if ("P-value" %in% extras & !("sigma" %in% extras)) {
1167                Pval_b <- c()
1168                sign_Pvalue <- c()
1169                for (i in (r+2):(2*r+2)) {
1170                        b_ci[i,1] <- 2*Pr_z[i-r-1] - quantiles[i,2]
1171                        b_ci[i,2] <- 2*Pr_z[i-r-1] - quantiles[i,1]
1172                        statistic <- paste0("P-value(b",i-r-2,")")
1173                        Pval_b[i-r-1] <- statistic
1174                        b_ci <- cbind(b_ci[ ,1:2],statistic)
1175                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[i,1], b_ci[i,2]), sep="", "\n")
1176                        if (bar==TRUE) print(ggplot(b_ci[i, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1177                        +ggtitle(paste("Basic Confidence Interval for", statistic)))
1178                        if (coeff_Pvalue[i-r-1] >0.1) sign_Pvalue[i-r-1] <- " "
1179                        if (coeff_Pvalue[i-r-1] <=0.1 & coeff_Pvalue[i-r-1] >0.05) sign_Pvalue[i-r-1] <- "*"
1180                        if (coeff_Pvalue[i-r-1] <=0.05 & coeff_Pvalue[i-r-1] >0.01) sign_Pvalue[i-r-1] <- "**"
1181                        if (coeff_Pvalue[i-r-1] <=0.01) sign_Pvalue[i-r-1] <- "***"
1182                }
1183                b_ci[2*r+3,1] <- 2*sd_hat - quantiles[2*r+3,2]
1184                b_ci[2*r+3,2] <- 2*sd_hat - quantiles[2*r+3,1]
1185                statistic <- c("sd")
1186                b_ci <- cbind(b_ci[ ,1:2],statistic)
1187                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[2*r+3,1], b_ci[2*r+3,2]), sep="", "\n")
1188                if (bar==TRUE) print(ggplot(b_ci[2*r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
```

```
+ggtitle(paste("Basic Confidence Interval for", statistic)))
if ("R^2" %in% extras) {
        b_ci[2*r+4,1] <- 2*R_2 - quantiles[2*r+4,2]
        b_ci[2*r+4,2] <- 2*R_2 - quantiles[2*r+4,1]
        statistic <- c("R^2")
        b_ci <- cbind(b_ci[ ,1:2],statistic)
        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[2*r+4,1], b_ci[2*r+4,2]), sep="", "\n")
        if (bar==TRUE) print(ggplot(b_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
        +ggtitle(paste("Basic Confidence Interval for", statistic)))
        if ("AIC" %in% extras) {
                b_ci[2*r+5,1] <- 2*AIC - quantiles[2*r+5,2]
                b_ci[2*r+5,2] <- 2*AIC - quantiles[2*r+5,1]
                statistic <- c("AIC")
                b_ci <- cbind(b_ci[ ,1:2],statistic)
                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                        b_ci[2*r+5,1], b_ci[2*r+5,2]), sep="", "\n")
                if (bar==TRUE) print(ggplot(b_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                +ggtitle(paste("Basic Confidence Interval for", statistic)))
                if ("BIC" %in% extras) {
                        b_ci[2*r+6,1] <- 2*BIC - quantiles[2*r+6,2]
                        b_ci[2*r+6,2] <- 2*BIC - quantiles[2*r+6,1]
                        statistic <- c("BIC")
                        b_ci <- cbind(b_ci[ ,1:2],statistic)
                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                                b_ci[2*r+6,1], b_ci[2*r+6,2]), sep="", "\n")
                        if (bar==TRUE) print(ggplot(b_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                        +ggtitle(paste("Basic Confidence Interval for", statistic)))
```

```
1216                        if ("RMSE" %in% extras) {
1217                                b_ci[2*r+7,1] <- 2*RMSE - quantiles[2*r+7,2]
1218                                b_ci[2*r+7,2] <- 2*RMSE - quantiles[2*r+7,1]
1219                                statistic <- c("RMSE")
1220                                b_ci <- cbind(b_ci[ ,1:2],statistic)
1221                                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1222                                        b_ci[2*r+7,1], b_ci[2*r+7,2]), sep="", "\n")
1223                                if (bar==TRUE) print(ggplot(b_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1224                                +ggtitle(paste("Basic Confidence Interval for", statistic)))
1225                        }
1226                }
1227        }
1228        if ( !("AIC" %in% extras)) {
1229                if ("BIC" %in% extras) {
1230                        b_ci[2*r+5,1] <- 2*BIC - quantiles[2*r+5,2]
1231                        b_ci[2*r+5,2] <- 2*BIC - quantiles[2*r+5,1]
1232                        statistic <- c("BIC")
1233                        b_ci <- cbind(b_ci[ ,1:2],statistic)
1234                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1235                                b_ci[2*r+5,1], b_ci[2*r+5,2]), sep="", "\n")
1236                        if (bar==TRUE) print(ggplot(b_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1237                        +ggtitle(paste("Basic Confidence Interval for", statistic)))
1238                        if ("RMSE" %in% extras) {
1239                                b_ci[2*r+6,1] <- 2*RMSE - quantiles[2*r+6,2]
1240                                b_ci[2*r+6,2] <- 2*RMSE - quantiles[2*r+6,1]
1241                                statistic <- c("RMSE")
1242                                b_ci <- cbind(b_ci[ ,1:2],statistic)
```

```
                                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                                                b_ci[2*r+6,1], b_ci[2*r+6,2]), sep="", "\n")
                                        if (bar==TRUE) print(ggplot(b_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                                        +ggtitle(paste("Basic Confidence Interval for", statistic)))
                                }
                        }
                        if ( !("BIC" %in% extras)) {
                                if ("RMSE" %in% extras) {
                                        b_ci[2*r+5,1] <- 2*RMSE - quantiles[2*r+5,2]
                                        b_ci[2*r+5,2] <- 2*RMSE - quantiles[2*r+5,1]
                                        statistic <- c("RMSE")
                                        b_ci <- cbind(b_ci[ ,1:2],statistic)
                                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                                                b_ci[2*r+5,1], b_ci[2*r+5,2]), sep="", "\n")
                                        if (bar==TRUE) print(ggplot(b_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                                        +ggtitle(paste("Basic Confidence Interval for", statistic)))
                                }
                        }
                }
        }
        if ( !("R^2" %in% extras)) {
                if ("AIC" %in% extras) {
                        b_ci[2*r+4,1] <- 2*AIC - quantiles[2*r+4,2]
                        b_ci[2*r+4,2] <- 2*AIC - quantiles[2*r+4,1]
                        statistic <- c("AIC")
                        b_ci <- cbind(b_ci[ ,1:2],statistic)
                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
```

```
1270                    b_ci[2*r+4,1], b_ci[2*r+4,2]), sep="", "\n")
1271                if (bar==TRUE) print(ggplot(b_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1272                +ggtitle(paste("Basic Confidence Interval for", statistic)))
1273                if ("BIC" %in% extras) {
1274                    b_ci[2*r+5,1] <- 2*BIC - quantiles[2*r+5,2]
1275                    b_ci[2*r+5,2] <- 2*BIC - quantiles[2*r+5,1]
1276                    statistic <- c("BIC")
1277                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1278                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1279                        b_ci[2*r+5,1], b_ci[2*r+5,2]), sep="", "\n")
1280                    if (bar==TRUE) print(ggplot(b_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1281                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1282                    if ("RMSE" %in% extras) {
1283                        b_ci[2*r+6,1] <- 2*RMSE - quantiles[2*r+6,2]
1284                        b_ci[2*r+6,2] <- 2*RMSE - quantiles[2*r+6,1]
1285                        statistic <- c("RMSE")
1286                        b_ci <- cbind(b_ci[ ,1:2],statistic)
1287                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1288                            b_ci[2*r+6,1], b_ci[2*r+6,2]), sep="", "\n")
1289                        if (bar==TRUE) print(ggplot(b_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1290                        +ggtitle(paste("Basic Confidence Interval for", statistic)))
1291                    }
1292                }
1293            }
1294            if ( !("AIC" %in% extras)) {
1295                if ("BIC" %in% extras) {
1296                    b_ci[2*r+4,1] <- 2*BIC - quantiles[2*r+4,2]
```

```
1297                    b_ci[2*r+4,2] <- 2*BIC - quantiles[2*r+4,1]
1298                    statistic <- c("BIC")
1299                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1300                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1301                            b_ci[2*r+4,1], b_ci[2*r+4,2]), sep="", "\n")
1302                    if (bar==TRUE) print(ggplot(b_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1303                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1304                    if ("RMSE" %in% extras) {
1305                            b_ci[2*r+5,1] <- 2*RMSE - quantiles[2*r+5,2]
1306                            b_ci[2*r+5,2] <- 2*RMSE - quantiles[2*r+5,1]
1307                            statistic <- c("RMSE")
1308                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1309                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1310                                    b_ci[2*r+5,1], b_ci[2*r+5,2]), sep="", "\n")
1311                            if (bar==TRUE) print(ggplot(b_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1312                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1313                    }
1314            }
1315            if ( !("BIC" %in% extras)) {
1316                    if ("RMSE" %in% extras) {
1317                            b_ci[2*r+4,1] <- 2*RMSE - quantiles[2*r+4,2]
1318                            b_ci[2*r+4,2] <- 2*RMSE - quantiles[2*r+4,1]
1319                            statistic <- c("RMSE")
1320                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1321                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1322                                    b_ci[2*r+4,1], b_ci[2*r+4,2]), sep="", "\n")
1323                            if (bar==TRUE) print(ggplot(b_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
```

```
                                        +ggtitle(paste("Basic Confidence Interval for", statistic)))
                                }
                        }
                }
        }

        if ( !("P-value" %in% extras) & !("sigma" %in% extras)) {
                b_ci[r+2,1] <- 2*sd_hat - quantiles[r+2,2]
                b_ci[r+2,2] <- 2*sd_hat - quantiles[r+2,1]
                statistic <- c("sd")
                b_ci <- cbind(b_ci[ ,1:2],statistic)
                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+2,1], b_ci[r+2,2]), sep="", "\n")
                if (bar==TRUE) print(ggplot(b_ci[r+2, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                +ggtitle(paste("Basic Confidence Interval for", statistic)))
                if ("R^2" %in% extras) {
                        b_ci[r+3,1] <- 2*R_2 - quantiles[r+3,2]
                        b_ci[r+3,2] <- 2*R_2 - quantiles[r+3,1]
                        statistic <- c("R^2")
                        b_ci <- cbind(b_ci[ ,1:2],statistic)
                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+3,1], b_ci[r+3,2]), sep="", "\n")
                        if (bar==TRUE) print(ggplot(b_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                        +ggtitle(paste("Basic Confidence Interval for", statistic)))
                        if ("AIC" %in% extras) {
                                b_ci[r+4,1] <- 2*AIC - quantiles[r+4,2]
                                b_ci[r+4,2] <- 2*AIC - quantiles[r+4,1]
                                statistic <- c("AIC")
```

```
1351                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1352                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1353                    if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1354                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1355                    if ("BIC" %in% extras) {
1356                            b_ci[r+5,1] <- 2*BIC - quantiles[r+5,2]
1357                            b_ci[r+5,2] <- 2*BIC - quantiles[r+5,1]
1358                            statistic <- c("BIC")
1359                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1360                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1361                                    b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
1362                            if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1363                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1364                            if ("RMSE" %in% extras) {
1365                                    b_ci[r+6,1] <- 2*RMSE - quantiles[r+6,2]
1366                                    b_ci[r+6,2] <- 2*RMSE - quantiles[r+6,1]
1367                                    statistic <- c("RMSE")
1368                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1369                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1370                                            b_ci[r+6,1], b_ci[r+6,2]), sep="", "\n")
1371                                    if (bar==TRUE) print(ggplot(b_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1372                                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1373                            }
1374                    }
1375            }
1376            if ( !("AIC" %in% extras)) {
1377                    if ("BIC" %in% extras) {
```

```
1378                b_ci[r+4,1] <- 2*BIC - quantiles[r+4,2]
1379                b_ci[r+4,2] <- 2*BIC - quantiles[r+4,1]
1380                statistic <- c("BIC")
1381                b_ci <- cbind(b_ci[ ,1:2],statistic)
1382                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1383                        b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1384                if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1385                +ggtitle(paste("Basic Confidence Interval for", statistic)))
1386                if ("RMSE" %in% extras) {
1387                        b_ci[r+5,1] <- 2*RMSE - quantiles[r+5,2]
1388                        b_ci[r+5,2] <- 2*RMSE - quantiles[r+5,1]
1389                        statistic <- c("RMSE")
1390                        b_ci <- cbind(b_ci[ ,1:2],statistic)
1391                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1392                                b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
1393                        if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1394                        +ggtitle(paste("Basic Confidence Interval for", statistic)))
1395                }
1396        }
1397        if ( !("BIC" %in% extras)) {
1398                if ("RMSE" %in% extras) {
1399                        b_ci[r+4,1] <- 2*RMSE - quantiles[r+4,2]
1400                        b_ci[r+4,2] <- 2*RMSE - quantiles[r+4,1]
1401                        statistic <- c("RMSE")
1402                        b_ci <- cbind(b_ci[ ,1:2],statistic)
1403                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1404                                b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
```

```
1405                                    if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1406                                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1407                                }
1408                            }
1409                        }
1410                    }
1411                    if ( !("R^2" %in% extras)) {
1412                        if ("AIC" %in% extras) {
1413                            b_ci[r+3,1] <- 2*AIC - quantiles[r+3,2]
1414                            b_ci[r+3,2] <- 2*AIC - quantiles[r+3,1]
1415                            statistic <- c("AIC")
1416                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1417                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+3,1], b_ci[r+3,2]), sep="", "\n")
1418                            if (bar==TRUE) print(ggplot(b_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1419                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1420                            if ("BIC" %in% extras) {
1421                                b_ci[r+4,1] <- 2*BIC - quantiles[r+4,2]
1422                                b_ci[r+4,2] <- 2*BIC - quantiles[r+4,1]
1423                                statistic <- c("BIC")
1424                                b_ci <- cbind(b_ci[ ,1:2],statistic)
1425                                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1426                                    b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1427                                if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1428                                +ggtitle(paste("Basic Confidence Interval for", statistic)))
1429                                if ("RMSE" %in% extras) {
1430                                    b_ci[r+5,1] <- 2*RMSE - quantiles[r+5,2]
1431                                    b_ci[r+5,2] <- 2*RMSE - quantiles[r+5,1]
```

```
1432                                    statistic <- c("RMSE")
1433                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1434                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1435                                            b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
1436                                    if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1437                                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1438                            }
1439                    }
1440            }
1441            if ( !("AIC" %in% extras)) {
1442                    if ("BIC" %in% extras) {
1443                            b_ci[r+3,1] <- 2*BIC - quantiles[r+3,2]
1444                            b_ci[r+3,2] <- 2*BIC - quantiles[r+3,1]
1445                            statistic <- c("BIC")
1446                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1447                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1448                                    b_ci[r+3,1], b_ci[r+3,2]), sep="", "\n")
1449                            if (bar==TRUE) print(ggplot(b_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1450                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1451                            if ("RMSE" %in% extras) {
1452                                    b_ci[r+4,1] <- 2*RMSE - quantiles[r+4,2]
1453                                    b_ci[r+4,2] <- 2*RMSE - quantiles[r+4,1]
1454                                    statistic <- c("RMSE")
1455                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1456                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1457                                            b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1458                                    if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
```

```
1459                                                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1460                                            }
1461                                    }
1462                                    if ( !("BIC" %in% extras)) {
1463                                            if ("RMSE" %in% extras) {
1464                                                    b_ci[r+3,1] <- 2*RMSE - quantiles[r+3,2]
1465                                                    b_ci[r+3,2] <- 2*RMSE - quantiles[r+3,1]
1466                                                    statistic <- c("RMSE")
1467                                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1468                                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1469                                                            b_ci[r+3,1], b_ci[r+3,2]), sep="", "\n")
1470                                                    if (bar==TRUE) print(ggplot(b_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1471                                                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1472                                            }
1473                                    }
1474                            }
1475                    }
1476            }
1477    }
1478
1479    Table <- cbind( c("C", colnames(z)))
1480    if ("sigma" %in% extras) Table <- cbind(c(Table, "sigma"))
1481    if ("P-value" %in% extras) Table <- cbind(c(Table, Pval_b))
1482    Table <- cbind(c(Table, "sd"))
1483    if ("R^2" %in% extras) Table <- cbind(c(Table, "R^2"))
1484    if ("AIC" %in% extras) Table <- cbind(c(Table, "AIC"))
1485    if ("BIC" %in% extras) Table <- cbind(c(Table, "BIC"))
```

```
1486        if ("RMSE" %in% extras) Table <- cbind(c(Table, "RMSE"))
1487
1488        if (ci=="basic") {
1489                Table <- cbind(Table,
1490                c(coeff_beta, if ("sigma" %in% extras) round(sum(bootstrap_sigma)/L,5), if ("P-value" %in% extras) coeff_Pvalue,
1491                        round(sum(bootstrap_sd)/L,5), if ("R^2" %in% extras) round(sum(bootstrap_R_2)/L,5),
1492                        if ("AIC" %in% extras) round(sum(bootstrap_AIC)/L,5), if ("BIC" %in% extras) round(sum(bootstrap_BIC)/L,5),
1493                        if ("RMSE" %in% extras) round(sum(bootstrap_RMSE)/L,5)),
1494                round(b_ci[ ,1],5),
1495                round(b_ci[ ,2],5),
1496                c(ifelse(b_ci[1:(r+1),1]<0 & b_ci[1:(r+1),2]>0, "X", "!"),
1497                        if ("sigma" %in% extras) c(" "),
1498                        if ("P-value" %in% extras) sign_Pvalue,
1499                        c(" "),
1500                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==4) c(" "," "," "," "),
1501                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==3) c(" "," "," "),
1502                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==2) c(" "," "),
1503                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==1) c(" "))
1504                )
1505                colnames(Table) <- c("Variable","Coefficient","Lower Basic CI","Upper Basic CI","Statistical Significance")
1506                View(Table)
1507        }
1508
1509        if (ci=="percentile") {
1510                Table <- cbind(Table,
1511                c(coeff_beta, if ("sigma" %in% extras) round(sum(bootstrap_sigma)/L,5), if ("P-value" %in% extras) coeff_Pvalue,
1512                        round(sum(bootstrap_sd)/L,5), if ("R^2" %in% extras) round(sum(bootstrap_R_2)/L,5),
```

```
                if ("AIC" %in% extras) round(sum(bootstrap_AIC)/L,5), if ("BIC" %in% extras) round(sum(bootstrap_BIC)/L,5),
                if ("RMSE" %in% extras) round(sum(bootstrap_RMSE)/L,5)),
            round(p_ci[ ,1],5),
            round(p_ci[ ,2],5),
            c(ifelse(p_ci[1:(r+1),1]<0 & p_ci[1:(r+1),2]>0, "X", "!"),
                if ("sigma" %in% extras) c(" "),
                if ("P-value" %in% extras) sign_Pvalue,
                c(" "),
                if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==4) c(" "," "," "," "),
                if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==3) c(" "," "," "),
                if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==2) c(" "," "),
                if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==1) c(" "))
            )
            colnames(Table) <- c("Variable","Coefficient","Lower Percentile CI","Upper Percentile CI","Statistical Significance")
            View(Table)
    }

    if (ci=="all") {
            Table <- cbind(Table,
            c(coeff_beta, if ("sigma" %in% extras) round(sum(bootstrap_sigma)/L,5), if ("P-value" %in% extras) coeff_Pvalue,
                round(sum(bootstrap_sd)/L,5), if ("R^2" %in% extras) round(sum(bootstrap_R_2)/L,5),
                if ("AIC" %in% extras) round(sum(bootstrap_AIC)/L,5), if ("BIC" %in% extras) round(sum(bootstrap_BIC)/L,5),
                if ("RMSE" %in% extras) round(sum(bootstrap_RMSE)/L,5)),
            round(p_ci[ ,1],5),
            round(p_ci[ ,2],5),
            c(ifelse(p_ci[1:(r+1),1]<0 & p_ci[1:(r+1),2]>0, "X", "!"),
                if ("sigma" %in% extras) c(" "),
```

```
                        if ("P-value" %in% extras) sign_Pvalue,
                        c(" "),
                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==4) c(" "," "," "," "),
                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==3) c(" "," "," "),
                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==2) c(" "," "),
                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==1) c(" ")),
                round(b_ci[ ,1],5),
                round(b_ci[ ,2],5),
                c(ifelse(b_ci[1:(r+1),1]<0 & b_ci[1:(r+1),2]>0, "X", "!"),
                        if ("sigma" %in% extras) c(" "),
                        if ("P-value" %in% extras) sign_Pvalue,
                        c(" "),
                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==4) c(" "," "," "," "),
                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==3) c(" "," "," "),
                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==2) c(" "," "),
                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==1) c(" "))
                )
                colnames(Table) <- c("Variable","Coefficient","Lower Percentile CI","Upper Percentile CI","Statistical Significance",
                        "Lower Basic CI","Upper Basic CI","Statistical Significance")
                View(Table)
        }
}
```

```
Algorithm_2 <- function(x, y, z, RTS="CRS", ORIENTATION="out", L1=100, L2=500, ci="percentile", alpha=0.05, extras=c(), seed=1, histogram=FALSE,
bar=FALSE) {

        if ("Benchmarking" %in% rownames(installed.packages()) == FALSE){
                install.packages("Benchmarking")
        }
        library(Benchmarking)

        if ("truncreg" %in% rownames(installed.packages()) == FALSE){
                install.packages("truncreg")
        }
        library(truncreg)

        if ("ggplot2" %in% rownames(installed.packages()) == FALSE){
                install.packages("ggplot2")
        }
        library(ggplot2)

        set.seed(seed)

        if (class(x)!="matrix") x <- as.matrix(x)
        if (class(y)!="matrix") y <- as.matrix(y)
        if (class(z)!="matrix") z <- as.matrix(z)

        if (length(colnames(z))== 0) colnames(z) <- paste0("z",1:ncol(z))

        while (alpha<=0 || alpha>=1) {
```

```
28              alpha <- readline("Choose a relatively small significance level betweeb 0 and 1. e.g.(0.05) ")
29      }
30      alpha <- as.numeric(alpha)
31
32      ## [1] DEA, computation of d^
33      dea <- dea(x, y, RTS = RTS, ORIENTATION = ORIENTATION)
34      d_hat <- eff(dea)
35
36      ## [2] Truncated Regression
37      r <- ncol(z)
38      auxiliary <- c()
39      beta_hat <- c()
40
41      if (ORIENTATION=="out") {
42              d_hat_t <- ifelse(d_hat > 1, d_hat, NA)
43              model_d_hat <- truncreg(d_hat_t ~ z, point = 1, direction = "left")
44      }
45
46      if (ORIENTATION=="in") {
47              d_hat_t <- ifelse(d_hat < 1, d_hat, NA)
48              model_d_hat <- truncreg(d_hat_t ~ z, point = 1, direction = "right")
49      }
50
51      bad <- is.na(d_hat_t)
52      non_na_d_hat_t <- d_hat_t[!bad]
53      trunc_resid <- non_na_d_hat_t - model_d_hat$fitted.values
54      sd_hat <- sd(trunc_resid)
```

```
55
56          for (i in 2:(r+1)) {
57          beta_hat[i-1] <- model_d_hat$coefficients[i]
58          }
59
60          ## [3] Bootstrap Loop 1
61          c <- c()
62          zb <- c()
63          n <- nrow(z)
64          d_star <- c()
65
66          ## [3.1] Drows from left-truncated normal distribution
67          for ( i in 1:n) {
68                  for (j in 1:r) {
69                          auxiliary[j] <- z[i,j]*beta_hat[j]
70                  }
71                  c[i] <- 1 - sum(auxiliary)
72                  zb[i] <- sum(auxiliary)
73          }
74
75          cc <- c/sd_hat
76          B <- matrix(nrow=n, ncol=L1)
77          for (j in 1:L1) {
78
79                  v <- runif(n, min=0, max=1)
80                  vv <- pnorm(cc) + (1 - pnorm(cc))*v
81                  u <- sd_hat*qnorm(vv)
```

```
82
83              ## [3.2] computation of d*
84              d_star <- zb + u
85
86              ## [3.3] setting x* and y*
87              x_star <- x
88              y_star <- y*d_hat/d_star
89
90              ## [3.4] DEA, computation of d^*
91              dea_2 <- dea(x_star, y_star, RTS = RTS, ORIENTATION = ORIENTATION)
92              d_hat_star <- eff(dea_2)
93
94              for (i in 1:n) {
95                      B[i,j] <- d_hat_star[i]
96              }
97      }
98      ## [4] computation of bias-corrected d^^
99      mean_di_hat_star <- c()
100
101     for (i in 1:nrow(B)) {
102             mean_di_hat_star[i] <- mean(B[i, ])
103     }
104
105     bias <- mean_di_hat_star - d_hat
106     d_2hat <- d_hat - bias
107
108     ## [5] estimation of b^^ and ó^^
```

```
beta_2hat <- c()

if (ORIENTATION=="out") {
        d_2hat_t <- ifelse(d_2hat > 1, d_2hat, NA)
        model_d_2hat <- truncreg(d_2hat_t ~ z, point = 1, direction = "left")
}

if (ORIENTATION=="in") {
        d_2hat_t <- ifelse(d_2hat < 1, d_2hat, NA)
        model_d_2hat <- truncreg(d_2hat_t ~ z, point = 1, direction = "right")
}

bad <- is.na(d_2hat_t)
non_na_d_2hat_t <- d_2hat_t[!bad]
trunc_resid <- non_na_d_2hat_t - model_d_2hat$fitted.values
sd_2hat <- sd(trunc_resid)
beta_2hat <- summary(model_d_2hat)$coefficients[1:(r+1),1]
if ("sigma" %in% extras) sigma <- summary(model_d_2hat)$coefficients[r+2,1]
if ("P-value" %in% extras & "sigma" %in% extras) Pr_z <- summary(model_d_2hat)$coefficients[1:(r+2),4]
if ("P-value" %in% extras & !("sigma" %in% extras)) Pr_z <- summary(model_d_2hat)$coefficients[1:(r+1),4]
if ("R^2" %in% extras) R_2 <- cor(model_d_2hat$fitted.values, non_na_d_2hat_t)^2
if ("AIC" %in% extras) AIC <- -2*model_d_2hat$logLik + 2*length(model_d_2hat$coefficients)
if ("BIC" %in% extras) BIC <- -2*model_d_2hat$logLik + log(length(model_d_2hat$fitted.values))*length(model_d_2hat$coefficients)
if ("RMSE" %in% extras) RMSE <- sqrt(sum(trunc_resid^2)/length(trunc_resid))

## [6] Bootstrap loop 2
bootstrap_beta = matrix(nrow=L2, ncol=r+1)
```

```
136         bootstrap_sd = c()
137         bootstrap_sigma = c()
138         bootstrap_R_2 = c()
139         if ("P-value" %in% extras & "sigma" %in% extras) bootstrap_pr_z = matrix(nrow=L2, ncol=r+2)
140         if ("P-value" %in% extras & !("sigma" %in% extras)) bootstrap_pr_z = matrix(nrow=L2, ncol=r+1)
141         if (!("P-value" %in% extras)) bootstrap_pr_z = c()
142         bootstrap_AIC = c()
143         bootstrap_BIC = c()
144         bootstrap_RMSE = c()
145         c_2 <- c()
146         zb_2 <- c()
147         d_2star <- c()
148         beta_2hat_star <- c()
149
150         for ( i in 1:n) {
151                 for (j in 1:r) {
152                         auxiliary[j] <- z[i,j]*beta_2hat[j+1]
153                 }
154                 c_2[i] <- 1 - sum(auxiliary)
155                 zb_2[i] <- sum(auxiliary)
156         }
157
158         cc_2 <- c_2/sd_2hat
159
160         for (j in 1:L2) {
161                 ## [6.1] Drows from left-truncated normal distribution
162                 v_2 <- runif(n, min=0, max=1)
```

```
vv_2 <- pnorm(cc_2) + (1 - pnorm(cc_2))*v_2
u_2 <- sd_2hat*qnorm(vv_2)

## [6.2] computation of d**
d_2star <- zb_2 + u_2

## [6.3] estimation of b^^* and ó^^*
if (ORIENTATION=="out") {
        d_2star_t <- ifelse(d_2star > 1, d_2star, NA)
        model_d_2star <- truncreg(d_2star_t ~ z, point = 1, direction = "left")
}

if (ORIENTATION=="in") {
        d_2star_t <- ifelse(d_2star < 1, d_2star, NA)
        model_d_2star <- truncreg(d_2star_t ~ z, point = 1, direction = "right")
}

bad <- is.na(d_2star_t)
non_na_d_2star_t <- d_2star_t[!bad]
trunc_resid <- non_na_d_2star_t - model_d_2star$fitted.values
sd_2hat_star <- sd(trunc_resid)
beta_2hat_star <- summary(model_d_2star)$coefficients[1:(r+1),1]
if ("sigma" %in% extras) sigma_star <- summary(model_d_2star)$coefficients[r+2,1]
if ("P-value" %in% extras & "sigma" %in% extras) Pr_z_star <- summary(model_d_2star)$coefficients[1:(r+2),4]
if ("P-value" %in% extras & !("sigma" %in% extras)) Pr_z_star <- summary(model_d_2star)$coefficients[1:(r+1),4]
if ("R^2" %in% extras) R_2_star <- cor(model_d_2star$fitted.values, non_na_d_2star_t)^2
if ("AIC" %in% extras) AIC_star <- -2*model_d_2star$logLik + 2*length(model_d_2star$coefficients)
```

```
190             if ("BIC" %in% extras) BIC_star <- -2*model_d_2star$logLik + log(length(model_d_2star$fitted.values))*length(model_d_2star$coefficients)
191             if ("RMSE" %in% extras) RMSE_star <- sqrt(sum(trunc_resid^2)/length(trunc_resid))
192
193             for (i in 1:(r+1)) {
194                     bootstrap_beta[j,i] <- beta_2hat_star[i]
195                     if ("P-value" %in% extras & !("sigma" %in% extras)) bootstrap_pr_z[j,i] <- Pr_z_star[i]
196             }
197             if ("P-value" %in% extras & "sigma" %in% extras) {
198                     for (i in 1:(r+2)) {
199                             if ("P-value" %in% extras) bootstrap_pr_z[j,i] <- Pr_z_star[i]
200                     }
201             }
202             bootstrap_sd[j] <- sd_2hat_star
203             if ("sigma" %in% extras) bootstrap_sigma[j] <- sigma_star
204             if ("R^2" %in% extras) bootstrap_R_2[j] <- R_2_star
205             if ("AIC" %in% extras) bootstrap_AIC[j] <- AIC_star
206             if ("BIC" %in% extras) bootstrap_BIC[j] <- BIC_star
207             if ("RMSE" %in% extras) bootstrap_RMSE[j] <- RMSE_star
208     }
209     C <- cbind(bootstrap_beta)
210     if ("sigma" %in% extras) C <- cbind(C, bootstrap_sigma)
211     if ("P-value" %in% extras) C <- cbind(C, bootstrap_pr_z)
212     C <- cbind(C, bootstrap_sd)
213     if ("R^2" %in% extras) C <- cbind(C, bootstrap_R_2)
214     if ("AIC" %in% extras) C <- cbind(C, bootstrap_AIC)
215     if ("BIC" %in% extras) C <- cbind(C, bootstrap_BIC)
216     if ("RMSE" %in% extras) C <- cbind(C, bootstrap_RMSE)
```

```
217
218         if (histogram==TRUE) {
219                 for (i in 1:ncol(C)) {
220                         hist(C[ ,i])
221                 }
222         }
223
224         ## [7] Confidence intervals
225         quantiles <- matrix(nrow=ncol(C), ncol=2)
226
227         for (i in 1:ncol(C)) {
228                 for (j in 1:2) {
229                         quantiles[i,j] <- quantile(C[ ,i],c(alpha/2,1-(alpha/2)))[j]
230                 }
231         }
232
233         bootstrap_beta <- as.data.frame(bootstrap_beta)
234         bootstrap_pr_z <- as.data.frame(bootstrap_pr_z)
235         coeff_beta <- round(sapply(bootstrap_beta, sum)/L2,5)
236         names(coeff_beta) <- NULL
237         if ("P-value" %in% extras) {
238                 coeff_Pvalue <- round(sapply(bootstrap_pr_z, sum)/L2,5)
239                 names(coeff_Pvalue) <- NULL
240         }
241
242         statistic <- c()
243         ## Percentile Confidence Intervals
```

```
244         if (ci=="percentile" || ci=="all") {
245             p_ci <- matrix(nrow=ncol(C), ncol=2)
246             p_ci <- as.data.frame(p_ci)
247             for (i in 1:(r+1)) {
248                 p_ci[i,1] <- quantiles[i,1]
249                 p_ci[i,2] <- quantiles[i,2]
250                 statistic <- paste0("b",i-1)
251                 p_ci <- cbind(p_ci[ ,1:2],statistic)
252                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[i,1], p_ci[i,2]), sep="", "\n")
253                 if (bar==TRUE) print(ggplot(p_ci[i, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
254                 +ggtitle(paste("Percentile Confidence Interval for", statistic))+geom_hline(yintercept=0, color="red",linetype = "longdash"))
255             }
256             if ("sigma" %in% extras) {
257                 p_ci[r+2,1] <- quantiles[r+2,1]
258                 p_ci[r+2,2] <- quantiles[r+2,2]
259                 statistic <- c("sigma")
260                 p_ci <- cbind(p_ci[ ,1:2],statistic)
261                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[r+2,1], p_ci[r+2,2]), sep="", "\n")
262                 if (bar==TRUE) print(ggplot(p_ci[r+2, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
263                 +ggtitle(paste("Percentile Confidence Interval for", statistic))+geom_hline(yintercept=0, color="red",linetype = "longdash"))
264                 if ("P-value" %in% extras) {
265                     Pval_b <- c()
266                     sign_Pvalue <- c()
267                     for (i in (r+3):(2*r+4)) {
268                         p_ci[i,1] <- quantiles[i,1]
269                         p_ci[i,2] <- quantiles[i,2]
270                         statistic <- paste0("P-value(b",i-r-3,")")
```

```
271                 Pval_b[i-r-2] <- statistic
272                 p_ci <- cbind(p_ci[ ,1:2],statistic)
273                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[i,1], p_ci[i,2]), sep="", "\n")
274                 if (bar==TRUE) print(ggplot(p_ci[i, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
275                 +ggtitle(paste("Percentile Confidence Interval for", statistic))
276                 +geom_hline(yintercept=0, color="red",linetype = "longdash"))
277                 if (coeff_Pvalue[i-r-2] >0.1) sign_Pvalue[i-r-2] <- " "
278                 if (coeff_Pvalue[i-r-2] <=0.1 & coeff_Pvalue[i-r-2] >0.05) sign_Pvalue[i-r-2] <- "*"
279                 if (coeff_Pvalue[i-r-2] <=0.05 & coeff_Pvalue[i-r-2] >0.01) sign_Pvalue[i-r-2] <- "**"
280                 if (coeff_Pvalue[i-r-2] <=0.01) sign_Pvalue[i-r-2] <- "***"
281         }
282         p_ci[2*r+5,1] <- quantiles[2*r+5,1]
283         p_ci[2*r+5,2] <- quantiles[2*r+5,2]
284         statistic <- c("sd")
285         p_ci <- cbind(p_ci[ ,1:2],statistic)
286         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[2*r+5,1], p_ci[2*r+5,2]), sep="", "\n")
287         if (bar==TRUE) print(ggplot(p_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
288         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
289         if ("R^2" %in% extras) {
290                 p_ci[2*r+6,1] <- quantiles[2*r+6,1]
291                 p_ci[2*r+6,2] <- quantiles[2*r+6,2]
292                 statistic <- c("R^2")
293                 p_ci <- cbind(p_ci[ ,1:2],statistic)
294                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
295                         p_ci[2*r+6,1], p_ci[2*r+6,2]), sep="", "\n")
296                 if (bar==TRUE) print(ggplot(p_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
297                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
```

```
298 if ("AIC" %in% extras) {
299         p_ci[2*r+7,1] <- quantiles[2*r+7,1]
300         p_ci[2*r+7,2] <- quantiles[2*r+7,2]
301         statistic <- c("AIC")
302         p_ci <- cbind(p_ci[ ,1:2],statistic)
303         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
304             p_ci[2*r+7,1], p_ci[2*r+7,2]), sep="", "\n")
305         if (bar==TRUE) print(ggplot(p_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
306         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
307         if ("BIC" %in% extras) {
308             p_ci[2*r+8,1] <- quantiles[2*r+8,1]
309             p_ci[2*r+8,2] <- quantiles[2*r+8,2]
310             statistic <- c("BIC")
311             p_ci <- cbind(p_ci[ ,1:2],statistic)
312             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
313                 p_ci[2*r+8,1], p_ci[2*r+8,2]), sep="", "\n")
314             if (bar==TRUE) print(ggplot(p_ci[2*r+8, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
315             +ggtitle(paste("Percentile Confidence Interval for", statistic)))
316             if ("RMSE" %in% extras) {
317                 p_ci[2*r+9,1] <- quantiles[2*r+9,1]
318                 p_ci[2*r+9,2] <- quantiles[2*r+9,2]
319                 statistic <- c("RMSE")
320                 p_ci <- cbind(p_ci[ ,1:2],statistic)
321                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f|",
322                     p_ci[2*r+9,1], p_ci[2*r+9,2]), sep="", "\n")
323                 if (bar==TRUE) print(ggplot(p_ci[2*r+9, ], aes(statistic, ymin = V1, ymax = V2))
324                 +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
```

```
                }
            }
        }
        if ( !("AIC" %in% extras)) {
            if ("BIC" %in% extras) {
                p_ci[2*r+7,1] <- quantiles[2*r+7,1]
                p_ci[2*r+7,2] <- quantiles[2*r+7,2]
                statistic <- c("BIC")
                p_ci <- cbind(p_ci[ ,1:2],statistic)
                cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                    p_ci[2*r+7,1], p_ci[2*r+7,2]), sep="", "\n")
                if (bar==TRUE) print(ggplot(p_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                +ggtitle(paste("Percentile Confidence Interval for", statistic)))
                if ("RMSE" %in% extras) {
                    p_ci[2*r+8,1] <- quantiles[2*r+8,1]
                    p_ci[2*r+8,2] <- quantiles[2*r+8,2]
                    statistic <- c("RMSE")
                    p_ci <- cbind(p_ci[ ,1:2],statistic)
                    cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                        p_ci[2*r+8,1], p_ci[2*r+8,2]), sep="", "\n")
                    if (bar==TRUE) print(ggplot(p_ci[2*r+8, ], aes(statistic, ymin = V1, ymax = V2))
                    +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
                }
            }
            if ( !("BIC" %in% extras)) {
                if ("RMSE" %in% extras) {
                    p_ci[2*r+7,1] <- quantiles[2*r+7,1]
```

```
352                                     p_ci[2*r+7,2] <- quantiles[2*r+7,2]
353                                     statistic <- c("RMSE")
354                                     p_ci <- cbind(p_ci[ ,1:2],statistic)
355                                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
356                                             p_ci[2*r+7,1], p_ci[2*r+7,2]), sep="", "\n")
357                                     if (bar==TRUE) print(ggplot(p_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))
358                                     +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
359                             }
360                     }
361             }
362     }
363     if ( !("R^2" %in% extras)) {
364             if ("AIC" %in% extras) {
365                     p_ci[2*r+6,1] <- quantiles[2*r+6,1]
366                     p_ci[2*r+6,2] <- quantiles[2*r+6,2]
367                     statistic <- c("AIC")
368                     p_ci <- cbind(p_ci[ ,1:2],statistic)
369                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
370                             p_ci[2*r+6,1], p_ci[2*r+6,2]), sep="", "\n")
371                     if (bar==TRUE) print(ggplot(p_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
372                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
373                     if ("BIC" %in% extras) {
374                             p_ci[2*r+7,1] <- quantiles[2*r+7,1]
375                             p_ci[2*r+7,2] <- quantiles[2*r+7,2]
376                             statistic <- c("BIC")
377                             p_ci <- cbind(p_ci[ ,1:2],statistic)
378                             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
```

```
379                                     p_ci[2*r+7,1], p_ci[2*r+7,2]), sep="", "\n")
380                             if (bar==TRUE) print(ggplot(p_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
381                             +ggtitle(paste("Percentile Confidence Interval for", statistic)))
382                             if ("RMSE" %in% extras) {
383                                     p_ci[2*r+8,1] <- quantiles[2*r+8,1]
384                                     p_ci[2*r+8,2] <- quantiles[2*r+8,2]
385                                     statistic <- c("RMSE")
386                                     p_ci <- cbind(p_ci[ ,1:2],statistic)
387                                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
388                                             p_ci[2*r+8,1], p_ci[2*r+8,2]), sep="", "\n")
389                                     if (bar==TRUE) print(ggplot(p_ci[2*r+8, ], aes(statistic, ymin = V1, ymax = V2))
390                                     +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
391                             }
392                     }
393             }
394             if ( !("AIC" %in% extras)) {
395                     if ("BIC" %in% extras) {
396                             p_ci[2*r+6,1] <- quantiles[2*r+6,1]
397                             p_ci[2*r+6,2] <- quantiles[2*r+6,2]
398                             statistic <- c("BIC")
399                             p_ci <- cbind(p_ci[ ,1:2],statistic)
400                             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
401                                     p_ci[2*r+6,1], p_ci[2*r+6,2]), sep="", "\n")
402                             if (bar==TRUE) print(ggplot(p_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
403                             +ggtitle(paste("Percentile Confidence Interval for", statistic)))
404                             if ("RMSE" %in% extras) {
405                                     p_ci[2*r+7,1] <- quantiles[2*r+7,1]
```

```
406                                         p_ci[2*r+7,2] <- quantiles[2*r+7,2]
407                                         statistic <- c("RMSE")
408                                         p_ci <- cbind(p_ci[ ,1:2],statistic)
409                                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
410                                             p_ci[2*r+7,1], p_ci[2*r+7,2]), sep="", "\n")
411                                         if (bar==TRUE) print(ggplot(p_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))
412                                         +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
413                                     }
414                                 }
415                                 if ( !("BIC" %in% extras)) {
416                                     if ("RMSE" %in% extras) {
417                                         p_ci[2*r+6,1] <- quantiles[2*r+6,1]
418                                         p_ci[2*r+6,2] <- quantiles[2*r+6,2]
419                                         statistic <- c("RMSE")
420                                         p_ci <- cbind(p_ci[ ,1:2],statistic)
421                                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
422                                             p_ci[2*r+6,1], p_ci[2*r+6,2]), sep="", "\n")
423                                         if (bar==TRUE) print(ggplot(p_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))
424                                         +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
425                                     }
426                                 }
427                             }
428                         }
429                 }
430
431                 if ( !("P-value" %in% extras)) {
432                     p_ci[r+3,1] <- quantiles[r+3,1]
```

```
433                 p_ci[r+3,2] <- quantiles[r+3,2]
434                 statistic <- c("sd")
435                 p_ci <- cbind(p_ci[ ,1:2],statistic)
436                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[r+3,1], p_ci[r+3,2]), sep="", "\n")
437                 if (bar==TRUE) print(ggplot(p_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
438                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
439                 if ("R^2" %in% extras) {
440                         p_ci[r+4,1] <- quantiles[r+4,1]
441                         p_ci[r+4,2] <- quantiles[r+4,2]
442                         statistic <- c("R^2")
443                         p_ci <- cbind(p_ci[ ,1:2],statistic)
444                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
445                                 p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
446                         if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))
447                         +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
448                         if ("AIC" %in% extras) {
449                                 p_ci[r+5,1] <- quantiles[r+5,1]
450                                 p_ci[r+5,2] <- quantiles[r+5,2]
451                                 statistic <- c("AIC")
452                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
453                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
454                                         p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
455                                 if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
456                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
457                                 if ("BIC" %in% extras) {
458                                         p_ci[r+6,1] <- quantiles[r+6,1]
459                                         p_ci[r+6,2] <- quantiles[r+6,2]
```

```
460                         statistic <- c("BIC")
461                         p_ci <- cbind(p_ci[ ,1:2],statistic)
462                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
463                                 p_ci[r+6,1], p_ci[r+6,2]), sep="", "\n")
464                         if (bar==TRUE) print(ggplot(p_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
465                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
466                         if ("RMSE" %in% extras) {
467                                 p_ci[r+7,1] <- quantiles[r+7,1]
468                                 p_ci[r+7,2] <- quantiles[r+7,2]
469                                 statistic <- c("RMSE")
470                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
471                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
472                                         p_ci[r+7,1], p_ci[r+7,2]), sep="", "\n")
473                                 if (bar==TRUE) print(ggplot(p_ci[r+7, ], aes(statistic, ymin = V1, ymax = V2))
474                                 +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
475                         }
476                 }
477         }
478         if ( !("AIC" %in% extras)) {
479                 if ("BIC" %in% extras) {
480                         p_ci[r+5,1] <- quantiles[r+5,1]
481                         p_ci[r+5,2] <- quantiles[r+5,2]
482                         statistic <- c("BIC")
483                         p_ci <- cbind(p_ci[ ,1:2],statistic)
484                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
485                                 p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
486                         if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
```

```
487                                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
488                                     if ("RMSE" %in% extras) {
489                                             p_ci[r+6,1] <- quantiles[r+6,1]
490                                             p_ci[r+6,2] <- quantiles[r+6,2]
491                                             statistic <- c("RMSE")
492                                             p_ci <- cbind(p_ci[ ,1:2],statistic)
493                                             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
494                                                     p_ci[r+6,1], p_ci[r+6,2]), sep="", "\n")
495                                             if (bar==TRUE) print(ggplot(p_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))
496                                             +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
497                                     }
498                             }
499                             if ( !("BIC" %in% extras)) {
500                                     if ("RMSE" %in% extras) {
501                                             p_ci[r+5,1] <- quantiles[r+5,1]
502                                             p_ci[r+5,2] <- quantiles[r+5,2]
503                                             statistic <- c("RMSE")
504                                             p_ci <- cbind(p_ci[ ,1:2],statistic)
505                                             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
506                                                     p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
507                                             if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))
508                                             +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
509                                     }
510                             }
511                     }
512             }
513             if ( !("R^2" %in% extras)) {
```

```
514                 if ("AIC" %in% extras) {
515                         p_ci[r+4,1] <- quantiles[r+4,1]
516                         p_ci[r+4,2] <- quantiles[r+4,2]
517                         statistic <- c("AIC")
518                         p_ci <- cbind(p_ci[ ,1:2],statistic)
519                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f|",
520                                 p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
521                         if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
522                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
523                         if ("BIC" %in% extras) {
524                                 p_ci[r+5,1] <- quantiles[r+5,1]
525                                 p_ci[r+5,2] <- quantiles[r+5,2]
526                                 statistic <- c("BIC")
527                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
528                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
529                                         p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
530                                 if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
531                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
532                                 if ("RMSE" %in% extras) {
533                                         p_ci[r+6,1] <- quantiles[r+6,1]
534                                         p_ci[r+6,2] <- quantiles[r+6,2]
535                                         statistic <- c("RMSE")
536                                         p_ci <- cbind(p_ci[ ,1:2],statistic)
537                                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
538                                                 p_ci[r+6,1], p_ci[r+6,2]), sep="", "\n")
539                                         if (bar==TRUE) print(ggplot(p_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))
540                                         +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
```

```
541                                     }
542                             }
543             }
544             if ( !("AIC" %in% extras)) {
545                     if ("BIC" %in% extras) {
546                             p_ci[r+4,1] <- quantiles[r+4,1]
547                             p_ci[r+4,2] <- quantiles[r+4,2]
548                             statistic <- c("BIC")
549                             p_ci <- cbind(p_ci[ ,1:2],statistic)
550                             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
551                                     p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
552                             if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
553                             +ggtitle(paste("Percentile Confidence Interval for", statistic)))
554                             if ("RMSE" %in% extras) {
555                                     p_ci[r+5,1] <- quantiles[r+5,1]
556                                     p_ci[r+5,2] <- quantiles[r+5,2]
557                                     statistic <- c("RMSE")
558                                     p_ci <- cbind(p_ci[ ,1:2],statistic)
559                                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
560                                             p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
561                                     if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))
562                                     +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
563                             }
564                     }
565                     if ( !("BIC" %in% extras)) {
566                             if ("RMSE" %in% extras) {
567                                     p_ci[r+4,1] <- quantiles[r+4,1]
```

```
                                                    p_ci[r+4,2] <- quantiles[r+4,2]
                                                    statistic <- c("RMSE")
                                                    p_ci <- cbind(p_ci[ ,1:2],statistic)
                                                    cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                                                            p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
                                                    if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))
                                                    +geom_errorbar()+ggtitle(paste("Percentile Confidence Interval for", statistic)))
                                            }
                                    }
                            }
                    }
            }

if ("P-value" %in% extras & !("sigma" %in% extras)) {
        Pval_b <- c()
        sign_Pvalue <- c()
        for (i in (r+2):(2*r+2)) {
                p_ci[i,1] <- quantiles[i,1]
                p_ci[i,2] <- quantiles[i,2]
                statistic <- paste0("P-value(b",i-r-2,")")
                Pval_b[i-r-1] <- statistic
                p_ci <- cbind(p_ci[ ,1:2],statistic)
                cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[i,1], p_ci[i,2]), sep="", "\n")
                if (bar==TRUE) print(ggplot(p_ci[i, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                +ggtitle(paste("Percentile Confidence Interval for", statistic)))
                if (coeff_Pvalue[i-r-1] >0.1) sign_Pvalue[i-r-1] <- " "
```

```
595             if (coeff_Pvalue[i-r-1] <=0.1 & coeff_Pvalue[i-r-1] >0.05) sign_Pvalue[i-r-1] <- "*"
596             if (coeff_Pvalue[i-r-1] <=0.05 & coeff_Pvalue[i-r-1] >0.01) sign_Pvalue[i-r-1] <- "**"
597             if (coeff_Pvalue[i-r-1] <=0.01) sign_Pvalue[i-r-1] <- "***"
598         }
599         p_ci[2*r+3,1] <- quantiles[2*r+3,1]
600         p_ci[2*r+3,2] <- quantiles[2*r+3,2]
601         statistic <- c("sd")
602         p_ci <- cbind(p_ci[ ,1:2],statistic)
603         cat("Percentile Confidence Interval for", " ". statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[2*r+3,1], p_ci[2*r+3,2]), sep="", "\n")
604         if (bar==TRUE) print(ggplot(p_ci[2*r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
605         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
606         if ("R^2" %in% extras) {
607             p_ci[2*r+4,1] <- quantiles[2*r+4,1]
608             p_ci[2*r+4,2] <- quantiles[2*r+4,2]
609             statistic <- c("R^2")
610             p_ci <- cbind(p_ci[ ,1:2],statistic)
611             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[2*r+4,1], p_ci[2*r+4,2]), sep="", "\n")
612             if (bar==TRUE) print(ggplot(p_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
613             +ggtitle(paste("Percentile Confidence Interval for", statistic)))
614             if ("AIC" %in% extras) {
615                 p_ci[2*r+5,1] <- quantiles[2*r+5,1]
616                 p_ci[2*r+5,2] <- quantiles[2*r+5,2]
617                 statistic <- c("AIC")
618                 p_ci <- cbind(p_ci[ ,1:2],statistic)
619                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
620                     p_ci[2*r+5,1], p_ci[2*r+5,2]), sep="", "\n")
621                 if (bar==TRUE) print(ggplot(p_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
```

```
622                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
623                 if ("BIC" %in% extras) {
624                         p_ci[2*r+6,1] <- quantiles[2*r+6,1]
625                         p_ci[2*r+6,2] <- quantiles[2*r+6,2]
626                         statistic <- c("BIC")
627                         p_ci <- cbind(p_ci[ ,1:2],statistic)
628                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
629                                 p_ci[2*r+6,1], p_ci[2*r+6,2]), sep="", "\n")
630                         if (bar==TRUE) print(ggplot(p_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
631                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
632                         if ("RMSE" %in% extras) {
633                                 p_ci[2*r+7,1] <- quantiles[2*r+7,1]
634                                 p_ci[2*r+7,2] <- quantiles[2*r+7,2]
635                                 statistic <- c("RMSE")
636                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
637                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
638                                         p_ci[2*r+7,1], p_ci[2*r+7,2]), sep="", "\n")
639                                 if (bar==TRUE) print(ggplot(p_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
640                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
641                         }
642                 }
643         }
644         if ( !("AIC" %in% extras)) {
645                 if ("BIC" %in% extras) {
646                         p_ci[2*r+5,1] <- quantiles[2*r+5,1]
647                         p_ci[2*r+5,2] <- quantiles[2*r+5,2]
648                         statistic <- c("BIC")
```

```
649                 p_ci <- cbind(p_ci[ ,1:2],statistic)
650                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
651                         p_ci[2*r+5,1], p_ci[2*r+5,2]), sep="", "\n")
652                 if (bar==TRUE) print(ggplot(p_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
653                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
654                 if ("RMSE" %in% extras) {
655                         p_ci[2*r+6,1] <- quantiles[2*r+6,1]
656                         p_ci[2*r+6,2] <- quantiles[2*r+6,2]
657                         statistic <- c("RMSE")
658                         p_ci <- cbind(p_ci[ ,1:2],statistic)
659                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
660                                 p_ci[2*r+6,1], p_ci[2*r+6,2]), sep="", "\n")
661                         if (bar==TRUE) print(ggplot(p_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
662                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
663                 }
664         }
665         if ( !("BIC" %in% extras)) {
666                 if ("RMSE" %in% extras) {
667                         p_ci[2*r+5,1] <- quantiles[2*r+5,1]
668                         p_ci[2*r+5,2] <- quantiles[2*r+5,2]
669                         statistic <- c("RMSE")
670                         p_ci <- cbind(p_ci[ ,1:2],statistic)
671                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
672                                 p_ci[2*r+5,1], p_ci[2*r+5,2]), sep="", "\n")
673                         if (bar==TRUE) print(ggplot(p_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
674                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
675                 }
```

```
676                         }
677                 }
678         }
679         if ( !("R^2" %in% extras)) {
680                 if ("AIC" %in% extras) {
681                         p_ci[2*r+4,1] <- quantiles[2*r+4,1]
682                         p_ci[2*r+4,2] <- quantiles[2*r+4,2]
683                         statistic <- c("AIC")
684                         p_ci <- cbind(p_ci[ ,1:2],statistic)
685                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
686                                 p_ci[2*r+4,1], p_ci[2*r+4,2]), sep="", "\n")
687                         if (bar==TRUE) print(ggplot(p_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
688                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
689                         if ("BIC" %in% extras) {
690                                 p_ci[2*r+5,1] <- quantiles[2*r+5,1]
691                                 p_ci[2*r+5,2] <- quantiles[2*r+5,2]
692                                 statistic <- c("BIC")
693                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
694                                 cat("Percentile Confidence Interval for", " ". statistic, ": ", sprintf("[%.5f,  %.5f]",
695                                         p_ci[2*r+5,1], p_ci[2*r+5,2]), sep="", "\n")
696                                 if (bar==TRUE) print(ggplot(p_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
697                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
698                                 if ("RMSE" %in% extras) {
699                                         p_ci[2*r+6,1] <- quantiles[2*r+6,1]
700                                         p_ci[2*r+6,2] <- quantiles[2*r+6,2]
701                                         statistic <- c("RMSE")
702                                         p_ci <- cbind(p_ci[ ,1:2],statistic)
```

```
703                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
704                                     p_ci[2*r+6,1], p_ci[2*r+6,2]), sep="", "\n")
705                                 if (bar==TRUE) print(ggplot(p_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
706                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
707                             }
708                         }
709                 }
710                 if ( !("AIC" %in% extras)) {
711                         if ("BIC" %in% extras) {
712                             p_ci[2*r+4,1] <- quantiles[2*r+4,1]
713                             p_ci[2*r+4,2] <- quantiles[2*r+4,2]
714                             statistic <- c("BIC")
715                             p_ci <- cbind(p_ci[ ,1:2],statistic)
716                             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
717                                 p_ci[2*r+4,1], p_ci[2*r+4,2]), sep="", "\n")
718                             if (bar==TRUE) print(ggplot(p_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
719                             +ggtitle(paste("Percentile Confidence Interval for", statistic)))
720                             if ("RMSE" %in% extras) {
721                                 p_ci[2*r+5,1] <- quantiles[2*r+5,1]
722                                 p_ci[2*r+5,2] <- quantiles[2*r+5,2]
723                                 statistic <- c("RMSE")
724                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
725                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
726                                     p_ci[2*r+5,1], p_ci[2*r+5,2]), sep="", "\n")
727                                 if (bar==TRUE) print(ggplot(p_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
728                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
729                             }
```

```
730                                     }
731                                     if ( !("BIC" %in% extras)) {
732                                             if ("RMSE" %in% extras) {
733                                                     p_ci[2*r+4,1] <- quantiles[2*r+4,1]
734                                                     p_ci[2*r+4,2] <- quantiles[2*r+4,2]
735                                                     statistic <- c("RMSE")
736                                                     p_ci <- cbind(p_ci[ ,1:2],statistic)
737                                                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
738                                                             p_ci[2*r+4,1], p_ci[2*r+4,2]), sep="", "\n")
739                                                     if (bar==TRUE) print(ggplot(p_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
740                                                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
741                                             }
742                                     }
743                             }
744                     }
745             }
746
747             if ( !("P-value" %in% extras) & !("sigma" %in% extras)) {
748                     p_ci[r+2,1] <- quantiles[r+2,1]
749                     p_ci[r+2,2] <- quantiles[r+2,2]
750                     statistic <- c("sd")
751                     p_ci <- cbind(p_ci[ ,1:2],statistic)
752                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[r+2,1], p_ci[r+2,2]), sep="", "\n")
753                     if (bar==TRUE) print(ggplot(p_ci[r+2, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
754                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
755                     if ("R^2" %in% extras) {
756                             p_ci[r+3,1] <- quantiles[r+3,1]
```

```
757                 p_ci[r+3,2] <- quantiles[r+3,2]
758                 statistic <- c("R^2")
759                 p_ci <- cbind(p_ci[ ,1:2],statistic)
760                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", p_ci[r+3,1], p_ci[r+3,2]), sep="", "\n")
761                 if (bar==TRUE) print(ggplot(p_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
762                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
763                 if ("AIC" %in% extras) {
764                         p_ci[r+4,1] <- quantiles[r+4,1]
765                         p_ci[r+4,2] <- quantiles[r+4,2]
766                         statistic <- c("AIC")
767                         p_ci <- cbind(p_ci[ ,1:2],statistic)
768                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
769                                 p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
770                         if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
771                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
772                         if ("BIC" %in% extras) {
773                                 p_ci[r+5,1] <- quantiles[r+5,1]
774                                 p_ci[r+5,2] <- quantiles[r+5,2]
775                                 statistic <- c("BIC")
776                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
777                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
778                                         p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
779                                 if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
780                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
781                                 if ("RMSE" %in% extras) {
782                                         p_ci[r+6,1] <- quantiles[r+6,1]
783                                         p_ci[r+6,2] <- quantiles[r+6,2]
```

```
784                                 statistic <- c("RMSE")
785                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
786                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
787                                         p_ci[r+6,1], p_ci[r+6,2]), sep="", "\n")
788                                 if (bar==TRUE) print(ggplot(p_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
789                                 +ggtitle(paste("Percentile Confidence Interval for", statistic)))
790                         }
791                 }
792         }
793         if ( !("AIC" %in% extras)) {
794                 if ("BIC" %in% extras) {
795                         p_ci[r+4,1] <- quantiles[r+4,1]
796                         p_ci[r+4,2] <- quantiles[r+4,2]
797                         statistic <- c("BIC")
798                         p_ci <- cbind(p_ci[ ,1:2],statistic)
799                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
800                                 p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
801                         if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
802                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
803                         if ("RMSE" %in% extras) {
804                                 p_ci[r+5,1] <- quantiles[r+5,1]
805                                 p_ci[r+5,2] <- quantiles[r+5,2]
806                                 statistic <- c("RMSE")
807                                 p_ci <- cbind(p_ci[ ,1:2],statistic)
808                                 cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
809                                         p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
810                                 if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
```

```
811                                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
812                             }
813                     }
814                     if ( !("BIC" %in% extras)) {
815                             if ("RMSE" %in% extras) {
816                                     p_ci[r+4,1] <- quantiles[r+4,1]
817                                     p_ci[r+4,2] <- quantiles[r+4,2]
818                                     statistic <- c("RMSE")
819                                     p_ci <- cbind(p_ci[ ,1:2],statistic)
820                                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
821                                             p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
822                                     if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
823                                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
824                             }
825                     }
826             }
827     }
828     if ( !("R^2" %in% extras)) {
829             if ("AIC" %in% extras) {
830                     p_ci[r+3,1] <- quantiles[r+3,1]
831                     p_ci[r+3,2] <- quantiles[r+3,2]
832                     statistic <- c("AIC")
833                     p_ci <- cbind(p_ci[ ,1:2],statistic)
834                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
835                             p_ci[r+3,1], p_ci[r+3,2]), sep="", "\n")
836                     if (bar==TRUE) print(ggplot(p_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
837                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
```

```
838                     if ("BIC" %in% extras) {
839                             p_ci[r+4,1] <- quantiles[r+4,1]
840                             p_ci[r+4,2] <- quantiles[r+4,2]
841                             statistic <- c("BIC")
842                             p_ci <- cbind(p_ci[ ,1:2],statistic)
843                             cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
844                                     p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
845                             if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
846                             +ggtitle(paste("Percentile Confidence Interval for", statistic)))
847                             if ("RMSE" %in% extras) {
848                                     p_ci[r+5,1] <- quantiles[r+5,1]
849                                     p_ci[r+5,2] <- quantiles[r+5,2]
850                                     statistic <- c("RMSE")
851                                     p_ci <- cbind(p_ci[ ,1:2],statistic)
852                                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
853                                             p_ci[r+5,1], p_ci[r+5,2]), sep="", "\n")
854                                     if (bar==TRUE) print(ggplot(p_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
855                                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
856                             }
857                     }
858             }
859             if ( !("AIC" %in% extras)) {
860                     if ("BIC" %in% extras) {
861                             p_ci[r+3,1] <- quantiles[r+3,1]
862                             p_ci[r+3,2] <- quantiles[r+3,2]
863                             statistic <- c("BIC")
864                             p_ci <- cbind(p_ci[ ,1:2],statistic)
```

```
865                     cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
866                         p_ci[r+3,1], p_ci[r+3,2]), sep="", "\n")
867                     if (bar==TRUE) print(ggplot(p_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
868                     +ggtitle(paste("Percentile Confidence Interval for", statistic)))
869                     if ("RMSE" %in% extras) {
870                         p_ci[r+4,1] <- quantiles[r+4,1]
871                         p_ci[r+4,2] <- quantiles[r+4,2]
872                         statistic <- c("RMSE")
873                         p_ci <- cbind(p_ci[ ,1:2],statistic)
874                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
875                             p_ci[r+4,1], p_ci[r+4,2]), sep="", "\n")
876                         if (bar==TRUE) print(ggplot(p_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
877                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
878                     }
879                 }
880                 if ( !("BIC" %in% extras)) {
881                     if ("RMSE" %in% extras) {
882                         p_ci[r+3,1] <- quantiles[r+3,1]
883                         p_ci[r+3,2] <- quantiles[r+3,2]
884                         statistic <- c("RMSE")
885                         p_ci <- cbind(p_ci[ ,1:2],statistic)
886                         cat("Percentile Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
887                             p_ci[r+3,1], p_ci[r+3,2]), sep="", "\n")
888                         if (bar==TRUE) print(ggplot(p_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
889                         +ggtitle(paste("Percentile Confidence Interval for", statistic)))
890                     }
891                 }
```

```
892                                 }
893                         }
894                 }
895         }
896
897         ## Basic Confidence Intervals
898         if (ci=="basic" || ci=="all") {
899                 b_ci <- matrix(nrow=ncol(C), ncol=2)
900                 b_ci <- as.data.frame(b_ci)
901                 for (i in 1:(r+1)) {
902                         b_ci[i,1] <- 2*beta_2hat[i] - quantiles[i,2]
903                         b_ci[i,2] <- 2*beta_2hat[i] - quantiles[i,1]
904                         statistic <- paste0("b",i-1)
905                         b_ci <- cbind(b_ci[ ,1:2],statistic)
906                         cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[i,1], b_ci[i,2]), sep="", "\n")
907                         if (bar==TRUE) print(ggplot(b_ci[i, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
908                         +ggtitle(paste("Basic Confidence Interval for", statistic))+geom_hline(yintercept=0, color="red",linetype = "longdash"))
909                 }
910                 if ("sigma" %in% extras) {
911                         b_ci[r+2,1] <- 2*sigma - quantiles[r+2,2]
912                         b_ci[r+2,2] <- 2*sigma - quantiles[r+2,1]
913                         statistic <- c("sigma")
914                         b_ci <- cbind(b_ci[ ,1:2],statistic)
915                         cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+2,1], b_ci[r+2,2]), sep="", "\n")
916                         if (bar==TRUE) print(ggplot(b_ci[r+2, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
917                         +ggtitle(paste("Basic Confidence Interval for", statistic)))
918                         if ("P-value" %in% extras) {
```

```
Pval_b <- c()
sign_Pvalue <- c()
for (i in (r+3):(2*r+4)) {
        b_ci[i,1] <- 2*Pr_z[i-r-2] - quantiles[i,2]
        b_ci[i,2] <- 2*Pr_z[i-r-2] - quantiles[i,1]
        statistic <- paste0("P-value(b",i-r-3,")")
        Pval_b[i-r-2] <- statistic
        b_ci <- cbind(b_ci[ ,1:2],statistic)
        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[i,1], b_ci[i,2]), sep="", "\n")
        if (bar==TRUE) print(ggplot(b_ci[i, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
        +ggtitle(paste("Basic Confidence Interval for", statistic)))
        if (coeff_Pvalue[i-r-2] >0.1) sign_Pvalue[i-r-2] <- " "
        if (coeff_Pvalue[i-r-2] <=0.1 & coeff_Pvalue[i-r-2] >0.05) sign_Pvalue[i-r-2] <- "*"
        if (coeff_Pvalue[i-r-2] <=0.05 & coeff_Pvalue[i-r-2] >0.01) sign_Pvalue[i-r-2] <- "**"
        if (coeff_Pvalue[i-r-2] <=0.01) sign_Pvalue[i-r-2] <- "***"
}
b_ci[2*r+5,1] <- 2*sd_2hat - quantiles[2*r+5,2]
b_ci[2*r+5,2] <- 2*sd_2hat - quantiles[2*r+5,1]
statistic <- c("sd")
b_ci <- cbind(b_ci[ ,1:2],statistic)
cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[2*r+5,1], b_ci[2*r+5,2]), sep="", "\n")
if (bar==TRUE) print(ggplot(b_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
+ggtitle(paste("Basic Confidence Interval for", statistic)))
if ("R^2" %in% extras) {
        b_ci[2*r+6,1] <- 2*R_2 - quantiles[2*r+6,2]
        b_ci[2*r+6,2] <- 2*R_2 - quantiles[2*r+6,1]
        statistic <- c("R^2")
```

```
946                 b_ci <- cbind(b_ci[ ,1:2],statistic)
947                 cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
948                     b_ci[2*r+6,1], b_ci[2*r+6,2]), sep="", "\n")
949                 if (bar==TRUE) print(ggplot(b_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
950                 +ggtitle(paste("Basic Confidence Interval for", statistic)))
951                 if ("AIC" %in% extras) {
952                     b_ci[2*r+7,1] <- 2*AIC - quantiles[2*r+7,2]
953                     b_ci[2*r+7,2] <- 2*AIC - quantiles[2*r+7,1]
954                     statistic <- c("AIC")
955                     b_ci <- cbind(b_ci[ ,1:2],statistic)
956                     cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
957                         b_ci[2*r+7,1], b_ci[2*r+7,2]), sep="", "\n")
958                     if (bar==TRUE) print(ggplot(b_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
959                     +ggtitle(paste("Basic Confidence Interval for", statistic)))
960                     if ("BIC" %in% extras) {
961                         b_ci[2*r+8,1] <- 2*BIC - quantiles[2*r+8,2]
962                         b_ci[2*r+8,2] <- 2*BIC - quantiles[2*r+8,1]
963                         statistic <- c("BIC")
964                         b_ci <- cbind(b_ci[ ,1:2],statistic)
965                         cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
966                             b_ci[2*r+8,1], b_ci[2*r+8,2]), sep="", "\n")
967                         if (bar==TRUE) print(ggplot(b_ci[2*r+8, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
968                         +ggtitle(paste("Basic Confidence Interval for", statistic)))
969                         if ("RMSE" %in% extras) {
970                             b_ci[2*r+9,1] <- 2*RMSE - quantiles[2*r+9,2]
971                             b_ci[2*r+9,2] <- 2*RMSE - quantiles[2*r+9,1]
972                             statistic <- c("RMSE")
```

```
                                b_ci <- cbind(b_ci[ ,1:2],statistic)
                                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                                        b_ci[2*r+9,1], b_ci[2*r+9,2]), sep="", "\n")
                                if (bar==TRUE) print(ggplot(b_ci[2*r+9, ], aes(statistic, ymin = V1, ymax = V2))
                                +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
                        }
                }
        }
        if ( !("AIC" %in% extras)) {
                if ("BIC" %in% extras) {
                        b_ci[2*r+7,1] <- 2*BIC - quantiles[2*r+7,2]
                        b_ci[2*r+7,2] <- 2*BIC - quantiles[2*r+7,1]
                        statistic <- c("BIC")
                        b_ci <- cbind(b_ci[ ,1:2],statistic)
                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                                b_ci[2*r+7,1], b_ci[2*r+7,2]), sep="", "\n")
                        if (bar==TRUE) print(ggplot(b_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                        +ggtitle(paste("Basic Confidence Interval for", statistic)))
                                if ("RMSE" %in% extras) {
                                        b_ci[2*r+8,1] <- 2*RMSE - quantiles[2*r+8,2]
                                        b_ci[2*r+8,2] <- 2*RMSE - quantiles[2*r+8,1]
                                        statistic <- c("RMSE")
                                        b_ci <- cbind(b_ci[ ,1:2],statistic)
                                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                                                b_ci[2*r+8,1], b_ci[2*r+8,2]), sep="", "\n")
                                        if (bar==TRUE) print(ggplot(b_ci[2*r+8, ], aes(statistic, ymin = V1, ymax = V2))
                                        +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
```

```
1000                                    }
1001                                }
1002                            if ( !("BIC" %in% extras)) {
1003                                if ("RMSE" %in% extras) {
1004                                    b_ci[2*r+7,1] <- 2*RMSE - quantiles[2*r+7,2]
1005                                    b_ci[2*r+7,2] <- 2*RMSE - quantiles[2*r+7,1]
1006                                    statistic <- c("RMSE")
1007                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1008                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1009                                        b_ci[2*r+7,1], b_ci[2*r+7,2]), sep="", "\n")
1010                                    if (bar==TRUE) print(ggplot(b_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))
1011                                    +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1012                                }
1013                            }
1014                        }
1015                    }
1016                    if ( !("R^2" %in% extras)) {
1017                        if ("AIC" %in% extras) {
1018                            b_ci[2*r+6,1] <- 2*AIC - quantiles[2*r+6,2]
1019                            b_ci[2*r+6,2] <- 2*AIC - quantiles[2*r+6,1]
1020                            statistic <- c("AIC")
1021                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1022                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1023                                b_ci[2*r+6,1], b_ci[2*r+6,2]), sep="", "\n")
1024                            if (bar==TRUE) print(ggplot(b_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1025                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1026                            if ("BIC" %in% extras) {
```

```
1027                    b_ci[2*r+7,1] <- 2*BIC - quantiles[2*r+7,2]
1028                    b_ci[2*r+7,2] <- 2*BIC - quantiles[2*r+7,1]
1029                    statistic <- c("BIC")
1030                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1031                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1032                            b_ci[2*r+7,1], b_ci[2*r+7,2]), sep="", "\n")
1033                    if (bar==TRUE) print(ggplot(b_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1034                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1035                    if ("RMSE" %in% extras) {
1036                            b_ci[2*r+8,1] <- 2*RMSE - quantiles[2*r+8,2]
1037                            b_ci[2*r+8,2] <- 2*RMSE - quantiles[2*r+8,1]
1038                            statistic <- c("RMSE")
1039                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1040                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1041                                    b_ci[2*r+8,1], b_ci[2*r+8,2]), sep="", "\n")
1042                            if (bar==TRUE) print(ggplot(b_ci[2*r+8, ], aes(statistic, ymin = V1, ymax = V2))
1043                            +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1044                    }
1045            }
1046    }
1047    if ( !("AIC" %in% extras)) {
1048            if ("BIC" %in% extras) {
1049                    b_ci[2*r+6,1] <- 2*BIC - quantiles[2*r+6,2]
1050                    b_ci[2*r+6,2] <- 2*BIC - quantiles[2*r+6,1]
1051                    statistic <- c("BIC")
1052                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1053                    cat("Basic Confidence Interval for", " ". statistic, ": ". sprintf("[%.5f,  %.5f]",
```

```
1054                                    b_ci[2*r+6,1], b_ci[2*r+6,2]), sep="", "\n")
1055                            if (bar==TRUE) print(ggplot(b_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1056                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1057                            if ("RMSE" %in% extras) {
1058                                    b_ci[2*r+7,1] <- 2*RMSE - quantiles[2*r+7,2]
1059                                    b_ci[2*r+7,2] <- 2*RMSE - quantiles[2*r+7,1]
1060                                    statistic <- c("RMSE")
1061                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1062                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1063                                            b_ci[2*r+7,1], b_ci[2*r+7,2]), sep="", "\n")
1064                                    if (bar==TRUE) print(ggplot(b_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))
1065                                    +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1066                            }
1067                    }
1068                    if ( !("BIC" %in% extras)) {
1069                            if ("RMSE" %in% extras) {
1070                                    b_ci[2*r+6,1] <- 2*RMSE - quantiles[2*r+6,2]
1071                                    b_ci[2*r+6,2] <- 2*RMSE - quantiles[2*r+6,1]
1072                                    statistic <- c("RMSE")
1073                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1074                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1075                                            b_ci[2*r+6,1], b_ci[2*r+6,2]), sep="", "\n")
1076                                    if (bar==TRUE) print(ggplot(b_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))
1077                                    +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1078                            }
1079                    }
1080            }
```

```
1081                    }
1082            }
1083
1084            if ( !("P-value" %in% extras)) {
1085                    b_ci[r+3,1] <- 2*sd_2hat - quantiles[r+3,2]
1086                    b_ci[r+3,2] <- 2*sd_2hat - quantiles[r+3,1]
1087                    statistic <- c("sd")
1088                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1089                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+3,1], b_ci[r+3,2]), sep="", "\n")
1090                    if (bar==TRUE) print(ggplot(b_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1091                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1092                    if ("R^2" %in% extras) {
1093                            b_ci[r+4,1] <- 2*R_2 - quantiles[r+4,2]
1094                            b_ci[r+4,2] <- 2*R_2 - quantiles[r+4,1]
1095                            statistic <- c("R^2")
1096                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1097                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1098                            if (bar--TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1099                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1100                            if ("AIC" %in% extras) {
1101                                    b_ci[r+5,1] <- 2*AIC - quantiles[r+5,2]
1102                                    b_ci[r+5,2] <- 2*AIC - quantiles[r+5,1]
1103                                    statistic <- c("AIC")
1104                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1105                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1106                                            b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
1107                                    if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
```

```
1108                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1109                    if ("BIC" %in% extras) {
1110                            b_ci[r+6,1] <- 2*BIC - quantiles[r+6,2]
1111                            b_ci[r+6,2] <- 2*BIC - quantiles[r+6,1]
1112                            statistic <- c("BIC")
1113                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1114                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1115                                    b_ci[r+6,1], b_ci[r+6,2]), sep="", "\n")
1116                            if (bar==TRUE) print(ggplot(b_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1117                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1118                            if ("RMSE" %in% extras) {
1119                                    b_ci[r+7,1] <- 2*RMSE - quantiles[r+7,2]
1120                                    b_ci[r+7,2] <- 2*RMSE - quantiles[r+7,1]
1121                                    statistic <- c("RMSE")
1122                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1123                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1124                                            b_ci[r+7,1], b_ci[r+7,2]), sep="", "\n")
1125                                    if (bar==TRUE) print(ggplot(b_ci[r+7, ], aes(statistic, ymin = V1, ymax = V2))
1126                                    +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1127                            }
1128                    }
1129            }
1130            if ( !("AIC" %in% extras)) {
1131                    if ("BIC" %in% extras) {
1132                            b_ci[r+5,1] <- 2*BIC - quantiles[r+5,2]
1133                            b_ci[r+5,2] <- 2*BIC - quantiles[r+5,1]
1134                            statistic <- c("BIC")
```

```
1135                b_ci <- cbind(b_ci[ ,1:2],statistic)
1136                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1137                        b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
1138                if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1139                +ggtitle(paste("Basic Confidence Interval for", statistic)))
1140                if ("RMSE" %in% extras) {
1141                        b_ci[r+6,1] <- 2*RMSE - quantiles[r+6,2]
1142                        b_ci[r+6,2] <- 2*RMSE - quantiles[r+6,1]
1143                        statistic <- c("RMSE")
1144                        b_ci <- cbind(b_ci[ ,1:2],statistic)
1145                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1146                                b_ci[r+6,1], b_ci[r+6,2]), sep="", "\n")
1147                        if (bar==TRUE) print(ggplot(b_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))
1148                        +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1149                }
1150        }
1151        if ( !("BIC" %in% extras)) {
1152                if ("RMSE" %in% extras) {
1153                        b_ci[r+5,1] <- 2*BIC - quantiles[r+5,2]
1154                        b_ci[r+5,2] <- 2*BIC - quantiles[r+5,1]
1155                        statistic <- c("RMSE")
1156                        b_ci <- cbind(b_ci[ ,1:2],statistic)
1157                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1158                                b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
1159                        if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))
1160                        +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1161                }
```

```
1162                                    }
1163                            }
1164                    }
1165                    if ( !("R^2" %in% extras)) {
1166                            if ("AIC" %in% extras) {
1167                                    b_ci[r+4,1] <- 2*AIC - quantiles[r+4,2]
1168                                    b_ci[r+4,2] <- 2*AIC - quantiles[r+4,1]
1169                                    statistic <- c("AIC")
1170                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1171                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1172                                            b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1173                                    if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1174                                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1175                                    if ("BIC" %in% extras) {
1176                                            b_ci[r+5,1] <- 2*BIC - quantiles[r+5,2]
1177                                            b_ci[r+5,2] <- 2*BIC - quantiles[r+5,1]
1178                                            statistic <- c("BIC")
1179                                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1180                                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1181                                                    b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
1182                                            if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1183                                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1184                                            if ("RMSE" %in% extras) {
1185                                                    b_ci[r+6,1] <- 2*RMSE - quantiles[r+6,2]
1186                                                    b_ci[r+6,2] <- 2*RMSE - quantiles[r+6,1]
1187                                                    statistic <- c("RMSE")
1188                                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
```

```
1189                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1190                                        b_ci[r+6,1], b_ci[r+6,2]), sep="", "\n")
1191                                    if (bar==TRUE) print(ggplot(b_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))
1192                                    +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1193                            }
1194                    }
1195            }
1196            if ( !("AIC" %in% extras)) {
1197                    if ("BIC" %in% extras) {
1198                            b_ci[r+4,1] <- 2*BIC - quantiles[r+4,2]
1199                            b_ci[r+4,2] <- 2*BIC - quantiles[r+4,1]
1200                            statistic <- c("BIC")
1201                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1202                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1203                                b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1204                            if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1205                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1206                            if ("RMSE" %in% extras) {
1207                                    b_ci[r+5,1] <- 2*RMSE - quantiles[r+5,2]
1208                                    b_ci[r+5,2] <- 2*RMSE - quantiles[r+5,1]
1209                                    statistic <- c("RMSE")
1210                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1211                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1212                                        b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
1213                                    if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))
1214                                    +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1215                            }
```

```
1216                                    }
1217                                    if ( !("BIC" %in% extras)) {
1218                                            if ("RMSE" %in% extras) {
1219                                                    b_ci[r+4,1] <- 2*RMSE - quantiles[r+4,2]
1220                                                    b_ci[r+4,2] <- 2*RMSE - quantiles[r+4,1]
1221                                                    statistic <- c("RMSE")
1222                                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1223                                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1224                                                            b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1225                                                    if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))
1226                                                    +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1227                                            }
1228                                    }
1229                            }
1230                    }
1231            }
1232    }
1233
1234    if ("P-value" %in% extras & !("sigma" %in% extras)) {
1235            Pval_b <- c()
1236            sign_Pvalue <- c()
1237            for (i in (r+2):(2*r+2)) {
1238                    b_ci[i,1] <- 2*Pr_z[i-r-1] - quantiles[i,2]
1239                    b_ci[i,2] <- 2*Pr_z[i-r-1] - quantiles[i,1]
1240                    statistic <- paste0("P-value(b",i-r-2,")")
1241                    Pval_b[i-r-1] <- statistic
1242                    b_ci <- cbind(b_ci[ ,1:2],statistic)
```

```
1243                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[i,1], b_ci[i,2]), sep="", "\n")
1244                if (bar==TRUE) print(ggplot(b_ci[i, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1245                +ggtitle(paste("Basic Confidence Interval for", statistic)))
1246                if (coeff_Pvalue[i-r-1] >0.1) sign_Pvalue[i-r-1] <- " "
1247                if (coeff_Pvalue[i-r-1] <=0.1 & coeff_Pvalue[i-r-1] >0.05) sign_Pvalue[i-r-1] <- "*"
1248                if (coeff_Pvalue[i-r-1] <=0.05 & coeff_Pvalue[i-r-1] >0.01) sign_Pvalue[i-r-1] <- "**"
1249                if (coeff_Pvalue[i-r-1] <=0.01) sign_Pvalue[i-r-1] <- "***"
1250        }
1251        b_ci[2*r+3,1] <- 2*sd_2hat - quantiles[2*r+3,2]
1252        b_ci[2*r+3,2] <- 2*sd_2hat - quantiles[2*r+3,1]
1253        statistic <- c("sd")
1254        b_ci <- cbind(b_ci[ ,1:2],statistic)
1255        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[2*r+3,1], b_ci[2*r+3,2]), sep="", "\n")
1256        if (bar==TRUE) print(ggplot(b_ci[2*r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1257        +ggtitle(paste("Basic Confidence Interval for", statistic)))
1258        if ("R^2" %in% extras) {
1259                b_ci[2*r+4,1] <- 2*R_2 - quantiles[2*r+4,2]
1260                b_ci[2*r+4,2] <- 2*R_2 - quantiles[2*r+4,1]
1261                statistic <- c("R^2")
1262                b_ci <- cbind(b_ci[ ,1:2],statistic)
1263                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[2*r+4,1], b_ci[2*r+4,2]), sep="", "\n")
1264                if (bar==TRUE) print(ggplot(b_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1265                +ggtitle(paste("Basic Confidence Interval for", statistic)))
1266                if ("AIC" %in% extras) {
1267                        b_ci[2*r+5,1] <- 2*AIC - quantiles[2*r+5,2]
1268                        b_ci[2*r+5,2] <- 2*AIC - quantiles[2*r+5,1]
1269                        statistic <- c("AIC")
```

```
1270                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1271                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1272                            b_ci[2*r+5,1], b_ci[2*r+5,2]), sep="", "\n")
1273                    if (bar==TRUE) print(ggplot(b_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1274                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1275                    if ("BIC" %in% extras) {
1276                            b_ci[2*r+6,1] <- 2*BIC - quantiles[2*r+6,2]
1277                            b_ci[2*r+6,2] <- 2*BIC - quantiles[2*r+6,1]
1278                            statistic <- c("BIC")
1279                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1280                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1281                                    b_ci[2*r+6,1], b_ci[2*r+6,2]), sep="", "\n")
1282                            if (bar==TRUE) print(ggplot(b_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1283                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1284                            if ("RMSE" %in% extras) {
1285                                    b_ci[2*r+7,1] <- 2*RMSE - quantiles[2*r+7,2]
1286                                    b_ci[2*r+7,2] <- 2*RMSE - quantiles[2*r+7,1]
1287                                    statistic <- c("RMSE")
1288                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1289                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1290                                            b_ci[2*r+7,1], b_ci[2*r+7,2]), sep="", "\n")
1291                                    if (bar==TRUE) print(ggplot(b_ci[2*r+7, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1292                                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1293                            }
1294                    }
1295            }
1296            if ( l("AIC" %in% extras)) {
```

```
1297        if ("BIC" %in% extras) {
1298            b_ci[2*r+5,1] <- 2*BIC - quantiles[2*r+5,2]
1299            b_ci[2*r+5,2] <- 2*BIC - quantiles[2*r+5,1]
1300            statistic <- c("BIC")
1301            b_ci <- cbind(b_ci[ ,1:2],statistic)
1302            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1303                b_ci[2*r+5,1], b_ci[2*r+5,2]), sep="", "\n")
1304            if (bar==TRUE) print(ggplot(b_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1305            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1306            if ("RMSE" %in% extras) {
1307                b_ci[2*r+6,1] <- 2*RMSE - quantiles[2*r+6,2]
1308                b_ci[2*r+6,2] <- 2*RMSE - quantiles[2*r+6,1]
1309                statistic <- c("RMSE")
1310                b_ci <- cbind(b_ci[ ,1:2],statistic)
1311                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1312                    b_ci[2*r+6,1], b_ci[2*r+6,2]), sep="", "\n")
1313                if (bar==TRUE) print(ggplot(b_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1314                +ggtitle(paste("Basic Confidence Interval for", statistic)))
1315            }
1316        }
1317        if ( !("BIC" %in% extras)) {
1318            if ("RMSE" %in% extras) {
1319                b_ci[2*r+5,1] <- 2*RMSE - quantiles[2*r+5,2]
1320                b_ci[2*r+5,2] <- 2*RMSE - quantiles[2*r+5,1]
1321                statistic <- c("RMSE")
1322                b_ci <- cbind(b_ci[ ,1:2],statistic)
1323                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
```

```
                                        b_ci[2*r+5,1], b_ci[2*r+5,2]), sep="", "\n")
                                    if (bar==TRUE) print(ggplot(b_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
                            }
                    }
            }
    }
    if ( !("R^2" %in% extras)) {
            if ("AIC" %in% extras) {
                    b_ci[2*r+4,1] <- 2*AIC - quantiles[2*r+4,2]
                    b_ci[2*r+4,2] <- 2*AIC - quantiles[2*r+4,1]
                    statistic <- c("AIC")
                    b_ci <- cbind(b_ci[ ,1:2],statistic)
                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                            b_ci[2*r+4,1], b_ci[2*r+4,2]), sep="", "\n")
                    if (bar==TRUE) print(ggplot(b_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
                    if ("BIC" %in% extras) {
                            b_ci[2*r+5,1] <- 2*BIC - quantiles[2*r+5,2]
                            b_ci[2*r+5,2] <- 2*BIC - quantiles[2*r+5,1]
                            statistic <- c("BIC")
                            b_ci <- cbind(b_ci[ ,1:2],statistic)
                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                                    b_ci[2*r+5,1], b_ci[2*r+5,2]), sep="", "\n")
                            if (bar==TRUE) print(ggplot(b_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
                            if ("RMSE" %in% extras) {
```

```
1351                                    b_ci[2*r+6,1] <- 2*RMSE - quantiles[2*r+6,2]
1352                                    b_ci[2*r+6,2] <- 2*RMSE - quantiles[2*r+6,1]
1353                                    statistic <- c("RMSE")
1354                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1355                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1356                                            b_ci[2*r+6,1], b_ci[2*r+6,2]), sep="", "\n")
1357                                    if (bar==TRUE) print(ggplot(b_ci[2*r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1358                                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1359                            }
1360                    }
1361            }
1362            if ( !("AIC" %in% extras)) {
1363                    if ("BIC" %in% extras) {
1364                            b_ci[2*r+4,1] <- 2*BIC - quantiles[2*r+4,2]
1365                            b_ci[2*r+4,2] <- 2*BIC - quantiles[2*r+4,1]
1366                            statistic <- c("BIC")
1367                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1368                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1369                                    b_ci[2*r+4,1], b_ci[2*r+4,2]), sep="", "\n")
1370                            if (bar==TRUE) print(ggplot(b_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1371                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1372                            if ("RMSE" %in% extras) {
1373                                    b_ci[2*r+5,1] <- 2*RMSE - quantiles[2*r+5,2]
1374                                    b_ci[2*r+5,2] <- 2*RMSE - quantiles[2*r+5,1]
1375                                    statistic <- c("RMSE")
1376                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1377                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
```

```
1378                                                    b_ci[2*r+5,1], b_ci[2*r+5,2]), sep="", "\n")
1379                                            if (bar==TRUE) print(ggplot(b_ci[2*r+5, ], aes(statistic, ymin = V1, ymax = V2))
1380                                            +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1381                                    }
1382                            }
1383                            if ( !("BIC" %in% extras)) {
1384                                    if ("RMSE" %in% extras) {
1385                                            b_ci[2*r+4,1] <- 2*RMSE - quantiles[2*r+4,2]
1386                                            b_ci[2*r+4,2] <- 2*RMSE - quantiles[2*r+4,1]
1387                                            statistic <- c("RMSE")
1388                                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1389                                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1390                                                    b_ci[2*r+4,1], b_ci[2*r+4,2]), sep="", "\n")
1391                                            if (bar==TRUE) print(ggplot(b_ci[2*r+4, ], aes(statistic, ymin = V1, ymax = V2))
1392                                            +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
1393                                    }
1394                            }
1395                    }
1396            }
1397    }
1398
1399    if ( !("P-value" %in% extras) & !("sigma" %in% extras)) {
1400            b_ci[r+2,1] <- 2*sd_2hat - quantiles[r+2,2]
1401            b_ci[r+2,2] <- 2*sd_2hat - quantiles[r+2,1]
1402            statistic <- c("sd")
1403            b_ci <- cbind(b_ci[ ,1:2],statistic)
1404            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+2,1], b_ci[r+2,2]), sep="", "\n")
```

```
1405        if (bar==TRUE) print(ggplot(b_ci[r+2, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1406        +ggtitle(paste("Basic Confidence Interval for", statistic)))
1407        if ("R^2" %in% extras) {
1408                b_ci[r+3,1] <- 2*R_2 - quantiles[r+3,2]
1409                b_ci[r+3,2] <- 2*R_2 - quantiles[r+3,1]
1410                statistic <- c("R^2")
1411                b_ci <- cbind(b_ci[ ,1:2],statistic)
1412                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+3,1], b_ci[r+3,2]), sep="", "\n")
1413                if (bar==TRUE) print(ggplot(b_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1414                +ggtitle(paste("Basic Confidence Interval for", statistic)))
1415                if ("AIC" %in% extras) {
1416                        b_ci[r+4,1] <- 2*AIC - quantiles[r+4,2]
1417                        b_ci[r+4,2] <- 2*AIC - quantiles[r+4,1]
1418                        statistic <- c("AIC")
1419                        b_ci <- cbind(b_ci[ ,1:2],statistic)
1420                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1421                        if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1422                        +ggtitle(paste("Basic Confidence Interval for", statistic)))
1423                        if ("BIC" %in% extras) {
1424                                b_ci[r+5,1] <- 2*BIC - quantiles[r+5,2]
1425                                b_ci[r+5,2] <- 2*BIC - quantiles[r+5,1]
1426                                statistic <- c("BIC")
1427                                b_ci <- cbind(b_ci[ ,1:2],statistic)
1428                                cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1429                                        b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
1430                                if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1431                                +ggtitle(paste("Basic Confidence Interval for", statistic)))
```

```
1432                    if ("RMSE" %in% extras) {
1433                        b_ci[r+6,1] <- 2*RMSE - quantiles[r+6,2]
1434                        b_ci[r+6,2] <- 2*RMSE - quantiles[r+6,1]
1435                        statistic <- c("RMSE")
1436                        b_ci <- cbind(b_ci[ ,1:2],statistic)
1437                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1438                            b_ci[r+6,1], b_ci[r+6,2]), sep="", "\n")
1439                        if (bar==TRUE) print(ggplot(b_ci[r+6, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1440                        +ggtitle(paste("Basic Confidence Interval for", statistic)))
1441                    }
1442                }
1443            }
1444            if ( !("AIC" %in% extras)) {
1445                if ("BIC" %in% extras) {
1446                    b_ci[r+4,1] <- 2*BIC - quantiles[r+4,2]
1447                    b_ci[r+4,2] <- 2*BIC - quantiles[r+4,1]
1448                    statistic <- c("BIC")
1449                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1450                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1451                        b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1452                    if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1453                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1454                    if ("RMSE" %in% extras) {
1455                        b_ci[r+5,1] <- 2*RMSE - quantiles[r+5,2]
1456                        b_ci[r+5,2] <- 2*RMSE - quantiles[r+5,1]
1457                        statistic <- c("RMSE")
1458                        b_ci <- cbind(b_ci[ ,1:2],statistic)
```

```
                                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                                                b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
                                        if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))
                                        +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
                                }
                        }
                        if ( !("BIC" %in% extras)) {
                                if ("RMSE" %in% extras) {
                                        b_ci[r+4,1] <- 2*RMSE - quantiles[r+4,2]
                                        b_ci[r+4,2] <- 2*RMSE - quantiles[r+4,1]
                                        statistic <- c("RMSE")
                                        b_ci <- cbind(b_ci[ ,1:2],statistic)
                                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
                                                b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
                                        if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))
                                        +geom_errorbar()+ggtitle(paste("Basic Confidence Interval for", statistic)))
                                }
                        }
                }
        }
        if ( !("R^2" %in% extras)) {
                if ("AIC" %in% extras) {
                        b_ci[r+3,1] <- 2*AIC - quantiles[r+3,2]
                        b_ci[r+3,2] <- 2*AIC - quantiles[r+3,1]
                        statistic <- c("AIC")
                        b_ci <- cbind(b_ci[ ,1:2],statistic)
                        cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]", b_ci[r+3,1], b_ci[r+3,2]), sep="", "\n")
```

```
1486                    if (bar==TRUE) print(ggplot(b_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1487                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1488                    if ("BIC" %in% extras) {
1489                            b_ci[r+4,1] <- 2*BIC - quantiles[r+4,2]
1490                            b_ci[r+4,2] <- 2*BIC - quantiles[r+4,1]
1491                            statistic <- c("BIC")
1492                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1493                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1494                                    b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1495                            if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1496                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1497                            if ("RMSE" %in% extras) {
1498                                    b_ci[r+5,1] <- 2*RMSE - quantiles[r+5,2]
1499                                    b_ci[r+5,2] <- 2*RMSE - quantiles[r+5,1]
1500                                    statistic <- c("RMSE")
1501                                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1502                                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1503                                            b_ci[r+5,1], b_ci[r+5,2]), sep="", "\n")
1504                                    if (bar==TRUE) print(ggplot(b_ci[r+5, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1505                                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1506                            }
1507                    }
1508            }
1509            if ( !("AIC" %in% extras)) {
1510                    if ("BIC" %in% extras) {
1511                            b_ci[r+3,1] <- 2*BIC - quantiles[r+3,2]
1512                            b_ci[r+3,2] <- 2*BIC - quantiles[r+3,1]
```

```
1513                    statistic <- c("BIC")
1514                    b_ci <- cbind(b_ci[ ,1:2],statistic)
1515                    cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1516                            b_ci[r+3,1], b_ci[r+3,2]), sep="", "\n")
1517                    if (bar--TRUE) print(ggplot(b_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1518                    +ggtitle(paste("Basic Confidence Interval for", statistic)))
1519                    if ("RMSE" %in% extras) {
1520                            b_ci[r+4,1] <- 2*RMSE - quantiles[r+4,2]
1521                            b_ci[r+4,2] <- 2*RMSE - quantiles[r+4,1]
1522                            statistic <- c("RMSE")
1523                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1524                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1525                                    b_ci[r+4,1], b_ci[r+4,2]), sep="", "\n")
1526                            if (bar==TRUE) print(ggplot(b_ci[r+4, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1527                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
1528                    }
1529            }
1530            if ( !("BIC" %in% extras)) {
1531                    if ("RMSE" %in% extras) {
1532                            b_ci[r+3,1] <- 2*RMSE - quantiles[r+3,2]
1533                            b_ci[r+3,2] <- 2*RMSE - quantiles[r+3,1]
1534                            statistic <- c("RMSE")
1535                            b_ci <- cbind(b_ci[ ,1:2],statistic)
1536                            cat("Basic Confidence Interval for", " ", statistic, ": ", sprintf("[%.5f,  %.5f]",
1537                                    b_ci[r+3,1], b_ci[r+3,2]), sep="", "\n")
1538                            if (bar==TRUE) print(ggplot(b_ci[r+3, ], aes(statistic, ymin = V1, ymax = V2))+geom_errorbar()
1539                            +ggtitle(paste("Basic Confidence Interval for", statistic)))
```

```
1540                                                    }
1541                                                }
1542                                            }
1543                                        }
1544                                    }
1545                                }
1546
1547                                Table <- cbind( c("C", colnames(z)))
1548                                if ("sigma" %in% extras) Table <- cbind(c(Table, "sigma"))
1549                                if ("P-value" %in% extras) Table <- cbind(c(Table, Pval_b))
1550                                Table <- cbind(c(Table, "sd"))
1551                                if ("R^2" %in% extras) Table <- cbind(c(Table, "R^2"))
1552                                if ("AIC" %in% extras) Table <- cbind(c(Table, "AIC"))
1553                                if ("BIC" %in% extras) Table <- cbind(c(Table, "BIC"))
1554                                if ("RMSE" %in% extras) Table <- cbind(c(Table, "RMSE"))
1555
1556                                if (ci=="basic") {
1557                                        Table <- cbind(Table,
1558                                        c(coeff_beta, if ("sigma" %in% extras) round(sum(bootstrap_sigma)/L2,5), if ("P-value" %in% extras) coeff_Pvalue,
1559                                                round(sum(bootstrap_sd)/L2,5), if ("R^2" %in% extras) round(sum(bootstrap_R_2)/L2,5),
1560                                                if ("AIC" %in% extras) round(sum(bootstrap_AIC)/L2,5), if ("BIC" %in% extras) round(sum(bootstrap_BIC)/L2,5),
1561                                                if ("RMSE" %in% extras) round(sum(bootstrap_RMSE)/L2,5)),
1562                                        round(b_ci[ ,1],5),
1563                                        round(b_ci[ ,2],5),
1564                                        c(ifelse(b_ci[1:(r+1),1]<0 & b_ci[1:(r+1),2]>0, "X", "!"),
1565                                                if ("sigma" %in% extras) c(" "),
1566                                                if ("P-value" %in% extras) sign_Pvalue,
```

```
1567                    c(" "),
1568                    if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==4) c(" "," "," "," "),
1569                    if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==3) c(" "," "," "),
1570                    if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==2) c(" "," "),
1571                    if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==1) c(" "))
1572            )
1573            colnames(Table) <- c("Variable","Coefficient","Lower Basic CI","Upper Basic CI","Statistical Significance")
1574            View(Table)
1575    }
1576
1577    if (ci=="percentile") {
1578            Table <- cbind(Table,
1579            c(coeff_beta, if ("sigma" %in% extras) round(sum(bootstrap_sigma)/L2,5), if ("P-value" %in% extras) coeff_Pvalue,
1580                    round(sum(bootstrap_sd)/L2,5), if ("R^2" %in% extras) round(sum(bootstrap_R_2)/L2,5),
1581                    if ("AIC" %in% extras) round(sum(bootstrap_AIC)/L2,5), if ("BIC" %in% extras) round(sum(bootstrap_BIC)/L2,5),
1582                    if ("RMSE" %in% extras) round(sum(bootstrap_RMSE)/L2,5)),
1583            round(p_ci[ ,1],5),
1584            round(p_ci[ ,2],5),
1585            c(ifelse(p_ci[1:(r+1),1]<0 & p_ci[1:(r+1),2]>0, "X", "!"),
1586                    if ("sigma" %in% extras) c(" "),
1587                    if ("P-value" %in% extras) sign_Pvalue,
1588                    c(" "),
1589                    if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==4) c(" "," "," "," "),
1590                    if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==3) c(" "," "," "),
1591                    if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==2) c(" "," "),
1592                    if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==1) c(" "))
1593            )
```

```
1594            colnames(Table) <- c("Variable","Coefficient","Lower Percentile CI","Upper Percentile CI","Statistical Significance")
1595            View(Table)
1596    }
1597
1598    if (ci=="all") {
1599            Table <- cbind(Table,
1600            c(coeff_beta, if ("sigma" %in% extras) round(sum(bootstrap_sigma)/L2,5), if ("P-value" %in% extras) coeff_Pvalue,
1601                    round(sum(bootstrap_sd)/L2,5), if ("R^2" %in% extras) round(sum(bootstrap_R_2)/L2,5),
1602                    if ("AIC" %in% extras) round(sum(bootstrap_AIC)/L2,5), if ("BIC" %in% extras) round(sum(bootstrap_BIC)/L2,5),
1603                    if ("RMSE" %in% extras) round(sum(bootstrap_RMSE)/L2,5)),
1604            round(p_ci[ ,1],5),
1605            round(p_ci[ ,2],5),
1606            c(ifelse(p_ci[1:(r+1),1]<0 & p_ci[1:(r+1),2]>0, "X", "!"),
1607                    if ("sigma" %in% extras) c(" "),
1608                    if ("P-value" %in% extras) sign_Pvalue,
1609                    c(" "),
1610                    if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==4) c(" "," "," "," "),
1611                    if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==3) c(" "," "," "),
1612                    if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==2) c(" "," "),
1613                    if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==1) c(" ")),
1614            round(b_ci[ ,1],5),
1615            round(b_ci[ ,2],5),
1616            c(ifelse(b_ci[1:(r+1),1]<0 & b_ci[1:(r+1),2]>0, "X", "!"),
1617                    if ("sigma" %in% extras) c(" "),
1618                    if ("P-value" %in% extras) sign_Pvalue,
1619                    c(" "),
1620                    if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==4) c(" "," "," "," "),
```

```
1621                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==3) c(" "," "," "),
1622                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==2) c(" "," "),
1623                        if (length(which(c("R^2","AIC","BIC","RMSE") %in% extras == TRUE))==1) c(" "))
1624                )
1625                colnames(Table) <- c("Variable","Coefficient","Lower Percentile CI","Upper Percentile CI","Statistical Significance",
1626                "Lower Basic CI","Upper Basic CI","Statistical Significance")
1627                View(Table)
1628        }
1629 }
```

References

Aly, H.Y., Grabowski, R., Pasurka, C., Rangan, N., 1990. Technical, scale, and allocative efficiencies in U.S. banking: an empirical investigation. Review of Economics and Statistics 72, 211–218

Berger, A.N., Mester, L.J., 2003. Explaining the dramatic changes in performance of US banks: technological change deregulation and dynamic changes in competition. Journal of Financial Intermediation 12, 57–95

Charnes, A., Cooper, W.W., Rhodes, E., 1978. Measuring the inefficiency of decision making units. European Journal of Operational Research 2, 429–444

Coelli, T., Rao, D.S.P., Battese, G.E., 1998. An Introduction to Efficiency and Productivity Analysis. Kluwer Academic Publishers, Inc., Boston

Cooper W., Seiford M. and Zhu J., 2011. Handbook on Data Envelopment Analysis

Fare, R., 1988. Fundamentals of Production Theory. Springer, Berlin

Farrell, M.J., 1957. The measurement of productive efficiency. Journal of the Royal Statistical Society, Series A 120, 253–281

Gattoufi, S., Oral, M., Reisman, A., 2004. Data envelopment analysis literature: a bibliography update (1951–2001). Socio-Economic Planning Sciences 38, 159–229

Gijbels, I., Mammen, E., Park, B.U., Simar, L., 1999. On estimation of monotone and concave frontier functions. Journal of the American Statistical Association 94, 220–228

Halkos, G.E., Tzeremes, N.G., 2013. A Two-Stage Double Bootstrap DEA: The Case of the Top 25 European Football Clubs' Efficiency Levels. Managerial and Decision Economics 34 (2), 108-115

Hall, P., 1986. On the number of bootstrap simulations required to construct a confidence interval. The Annals of Statistics 14, 1453–1462

Honore´, B.E., Powell, J.L., 1994. Pairwise difference estimators of censored and truncated regression models. Journal of Econometrics 64, 241–278

Hughes, J.P., Mester, L.J., 2012. Efficiency in Banking: Theory, Practice, and Evidence. The Oxford Handbook of Banking

Jackson M., Fethi D., 2000. Evaluating the technical efficiency of Turkish commercial banks: An Application of DEA and Tobit Analysis. International DEA Symposium, University of Queensland, Brisbane, Australia, 2-4 July, 2000.

Keramidou, I., Mimis, A., 2011. n application of the double-bootstrap data envelopment analysis to investigate sources of efficiency in the Greek poultry sector. World's Poultry Science Journal 67 (4), 675-686

Kneip, A., Park, B.U., Simar, L., 1998. A note on the convergence of nonparametric DEA estimators for production efficiency scores. Econometric Theory 14, 783–793

Korostelev, A., Simar, L., Tsybakov, A.B., 1995. On estimation of monotone and convex boundaries. Publications des Instituts de Statistique des Universite´s de Paris 39, 3–18

Latruffe, L., Davidova, S., Balcombe, K., 2008. Application of a double bootstrap to investigation of determinants of technical efficiency of farms in Central Europe. Journal of Productivity Analysis 29 (2), 183-191

Lee, B.L., Worthington, A.C., 2014. Technical efficiency of mainstream airlines and low-cost carriers: New evidence using bootstrap data envelopment analysis truncated regression. Journal of Air Transport Management 38, 15-20

Lewbel, A., Linton, O., 2002. Nonparametric censored and truncated regression. Econometrica 70, 765–779

Puig-Junoy, J., 1998. Technical efficiency in the clinical management of critically ill patients. Health Economics 7 (3), 263-277

Shephard, R.W., 1970. Theory of Cost and Production Function. Princeton University Press, Princeton

Simar, L., Wilson, P.W., 2007. Estimation and inference in two-stage, semi-parametric models of production processes. Journal of Econometrics 136 (1), 31-64

Weill, L., 2004. Measuring cost efficiency in European banking: A comparison of frontier techniques. Source of the DocumentJournal of Productivity Analysis 21 (2), 133-152

Wijesiri, M., Viganò, L., Meoli, M., 2015. Efficiency of microfinance institutions in Sri Lanka: A two-stage double bootstrap DEA approach. Economic Modelling 47, 74-83