eat: An R Package for fitting Efficiency Analysis Trees

by Miriam Esteve, Victor España, Juan Aparicio, and Xavier Barber

Abstract eat is a new package for R that includes functions to estimate production frontiers and technical efficiency measures through non-parametric techniques based upon regression trees. The package specifically implements the main algorithms associated with a recently introduced methodology for estimating the efficiency of a set of decision-making units in Economics and Engineering through Machine Learning techniques, called Efficiency Analysis Trees (Esteve et al. 2020). The package includes code for estimating input- and output-oriented radial measures, input- and output-oriented Russell measures, the directional distance function and the weighted additive model, plotting graphical representations of the production frontier by tree structures, and determining rankings of importance of input variables in the analysis. Additionally, it includes the code to perform an adaptation of Random Forest in estimating technical efficiency. This paper describes the methodology and implementation of the functions, and reports numerical results using a real data base application.

1 Introduction

Efficiency analysis refers to the discipline of estimating production frontiers while measuring the efficiency of a set of observations, named Decision Making Units (DMUs), which use several inputs to produce several outputs. In the literature of Economics, Engineering and Operations Research, the estimation of production frontiers is a current topic of interest (see, for example, Arnaboldi, Azzone, and Giorgino 2014; Aparicio et al. 2017; O'Donnell et al. 2018). In this line, many models for estimating production frontiers have been developed, resorting to parametric and non-parametric approaches. In the non-parametric approach, a functional form does not need to be specified (e.g. a Cobb-Douglas production function) through the specification of a set of parameters to be estimated, since they are usually data-driven. Additionally, non-parametric models innately cope with multiple-output scenarios. In contrast, the parametric approach aggregates the outputs into a single production index or attempts to model the technology using a dual cost function (Orea and Zof 'io 2019). These are some of the advantages that makes the non-parametric approaches for measuring technical efficiency more appealing than their parametric counterparts.

Contextualizing the non-parametric measurement of efficiency analysis requires outlining the following works. Farrel (1957) was a renowned opponent of estimating efficiency by determining average performance and, indeed, he was the first author in the literature to introduce a method for constructing production frontiers as the maximum producible output from an input bundle. Inspired by Koopmans (1951) and Debreu (1951), Farrell introduced a piece-wise linear upper enveloping surface of the data cloud as the specification of the production frontier, satisfying some microeconomics postulates: free disposability, convexity and minimal extrapolation. A DMU is considered technically inefficient if it is located below the frontier. Furthermore, Farrell's measure of efficiency, inspired by Shephard (1953), is based on radial movements (equiproportional changes) from technically inefficient observations to their benchmarks located at the estimated production frontier. In the same context as Farrell, Afriat (1972) determined a production frontier under non-decreasing and concavity mathematical assumptions and, at the same time, as close as possible to the sample of observations. Finally, in the same line of research, Charnes, Cooper, and Rhodes (1978) and Banker, Charnes, and Cooper (1984) proposed Data Envelopment Analysis (DEA), rooted in mathematical programming to provide a relative efficiency assessment of a set of DMUs by the construction of a piece-wise linear frontier. Along with DEA, Free Disposal Hull (FDH) is another of the most recognized non-parametric models for estimating production frontiers. FDH is a deterministic model introduced by Deprins and Simar (1984), which is only based on the free disposability and minimal extrapolation principles, as opposed to DEA, which also assumes convexity. In fact, FDH can be considered the skeleton of DEA, since the convex hull of the former coincides with DEA's frontier (see Daraio and Simar 2005). In addition, other recent alternative non-parametric techniques for estimating production frontiers are: Banker and Maindiratta (1992) and Banker (1993), who showed that DEA can be interpreted as a Maximum Likelihood estimator; Simar and Wilson (1998, 2000; 2000), who introduced how to determine confidence intervals for the efficiency score of each DMU through adapting the bootstrapping methodology by Efron (1979) to the context of FDH and DEA; or Kuosmanen and Johnson (2010; 2017), who have recently shown that DEA may be interpreted as non-parametric least-squares regression, subject to shape constraints on the production frontier and sign constraints on residuals; to name a few.

However, few of the above methodologies are based upon Machine Learning techniques, despite being a rising research field (see, for example, the recent papers by Khezrimotlagh et al. 2019; and Zhu et al. 2019; or the book by Charles, Aparicio, and Zhu 2020). Recently, a bridge has been built between these literatures, Machine Learning and production theory, through a new technique proposed in Esteve et al. (2020), called Efficiency Analysis Trees. This new method shares some similarities with the standard FDH technique. In contrast to FDH, Efficiency Analysis Trees overcomes the well-known problem of overfitting linked to FDH and DEA, by using cross-validation to prune back the deep tree obtained in a first stage. Esteve et al. (2020) also showed that the performance of Efficiency Analysis Trees, checked through Monte Carlo simulations, clearly outperforms the FDH technique with respect to bias and mean squared error.

Many of the standard models for estimating technical efficiency are nowadays available as R packages such as: **Benchmarking** (Bogetoft and Otto 2010), for estimating technologies and measuring efficiencies using Data Envelopment Analysis (DEA) and Stochastic Frontier Analysis (SFA); **nonparaeff** (Oh and Suh 2013), for measuring efficiency and productivity using DEA and its variations; **npbr** (Daouia, Laurent, and Noh 2017), which covers data envelopment techniques based on piece-wise polynomials, splines, local linear fitting, extreme values and kernel smoothing; **snfa** (McKenzie 2018), which fits both a smooth analogue of DEA and a non-parametric analogue of SFA; or **semsfa** (Ferrara and Vidoli 2018), which, in a first stage, estimates Stochastic Frontier Models by semiparametric or non-parametric regression techniques to relax parametric restrictions and, in a second stage, applies a technique based on pseudolikelihood or the method of moments for estimating variance parameters. Additionally, there are other packages on efficiency measurement developed for alternative platforms. In MATLAB (The MathWorks Inc. 2021), we can find the **Data Envelopment Analysis Toolbox** (Álvarez, Barbero, and Zofio 2020), which implements the main DEA models and solves measures like the directional distance function (with desirable and undesirable outputs), the weighted additive model, and the Malmquist-Luenberger index; or the **Total Factor Productivity Toolbox** (Balk, Barbero, and Zof 'Io 2018), which includes functions to calculate the main Total Factor Productivity indices and their decomposition by DEA models. In Stata (StataCorp 2021), it is possible to find a similar package in Ji and Lee (2010).

In this paper, we introduce a new package in R, called **eat**, for fitting regression trees to estimate production frontiers in microeconomics and engineering, by implementing the main features of Efficiency Analysis Trees (Esteve et al. 2020). In particular, **eat** includes a complete set of baseline functions, covering a wide range of efficiency models fitted by Efficiency Analysis Trees (Esteve et al. 2020) and Random Forest (Esteve et al. 2021), and reporting numerical and graphical results. **eat** is available as free software, under the GNU General Public License version 3, and can be downloaded from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=eat`, including supplementary material as datasets or vignettes to replicate all the results presented in this paper. In addition, **eat** is hosted on an open source repository on GitHub at `https://github.com/MiriamEsteve/EAT`. The main objective of this package is the estimation of a production frontier through regression trees satisfying the microeconomic principles of free disposability, convexity and deterministic data. Free disposability states that if a certain input and output bundle is producible, then any input and output bundle that presents a greater value for inputs and a lower value for outputs is also producible. In some sense, it means that doing it worse is always feasible. Convexity means that if two input-output bundles are assumed producible, then any convex combination of them are also feasible. Finally, the deterministic quality means that the observations that belong to the data sample have been observed without noise. In other words, the technology always contains all these observations and, graphically, the production frontier envelops all the data cloud from above.

The efficiency measurement field has witnessed the introduction of many different technical efficiency measures throughout the last decades. Regarding the technical efficiency measures implemented in the new **eat** package, it is worth mentioning that numerical scores and barplots are provided for the output-oriented and input-oriented BCC radial models (Banker, Charnes, and Cooper 1984), the directional distance function (Chambers, Chung, and Färe 1998), the weighted additive model (Lovell and Pastor 1995; and W. W. Cooper, Park, and Pastor 1999) and the output-oriented and input-oriented Russell measures (Färe and Lovell 1978). Additionally, the adaptation of Random Forest (Breiman 2001) for dealing with ensembles of Efficiency Analysis Trees, recently introduced in Esteve et al. (2021) and denoted as RF+EAT, has been also incorporated into the new **eat** package. The frontier estimator based on Random Forest, which is associated with more robust results, also allows to determine out-of-sample efficiency evaluation for the assessed DMUs. Another remarkable aspect of Efficiency Analysis Trees is the inherited ability to calculate feature importance as performed by other tree-based models of Machine Learning. Specifically, this fact allows the researchers to know which are the most relevant variables for obtaining efficiency and thus getting an explanation of the level of technical efficiency identified for each assessed unit. This ranking of importance variable has been implemented in the **eat** package. Finally, and from a data visualization point of view, the obtained frontier from Efficiency Analysis Trees can be represented by means of a tree structure, ideal for high-dimensional

scenarios where the patterns between the inputs and the efficient levels of outputs are very complex. In addition, FDH and DEA standard models have been included in the new package in order to facilitate comparison with the efficiency scores determined by the Efficiency Analysis Trees technique. Also, the convexification of the estimation of the technology provided by Efficiency Analysis Trees, named Convexified Efficiency Analysis Trees (CEAT) by Aparicio et al. (2021), is implemented in the **eat** package, with the objective of determining estimations under the axiom of convexity.

The functions included in the **eat** package are summarized in Table 1. This table comprises two columns divided into four subsections for Efficiency Analysis Trees, Random Forest for Efficiency Analysis Trees, Convexified Efficiency Analysis Trees and functions for data simulation. The first column is the name of the main functions and the second one is the description of the functions and the reference of the paper in which we can find the most detailed theoretical explanation of the corresponding function.

The paper is organized as follows. The following section summarises the two methodologies implemented in the **eat** package in R: Efficiency Analysis Trees and Random Forest for Efficiency Analysis Trees. Section Data Structure describes the data structures that characterize the production possibility sets, the structure of the functions, the results, etc., and briefly explains which data are used to illustrate the package. Section Basic functions of the library presents the basic methods. The next Section Basic EAT and RFEAT models deals with the measurement of economic efficiency of FDH, DEA, Efficiency Analysis Trees, Random Forest for Efficiency Analysis Trees and Convexified Efficiency Analysis Trees models. Advanced options, including displaying and exporting results can be found in Section Advanced options and displaying and exporting results. Section Conclusions concludes.

Table 1: eat package functions.

Function	Description
Subsection 1: Efficiency Analysis Trees	
`EAT`	It generates a pruned Efficiency Analysis Trees model and returns an `EAT` object.
`bestEAT`	It computes the root mean squared error (RMSE) for a set of Efficiency Analysis Trees models made up of a set of user-entered hyperparameters. These models are fitted with a training sample and evaluated with a test sample.
`efficiencyEAT`	It computes the efficiency scores of a set of DMUs through an Efficiency Analysis Trees model and returns a `data.frame`. The FDH scores can also be computed. Alternative mathematical programming models for calculating the efficiency scores are: • `"BCC.OUT"`: The output-oriented BCC radial model. • `"BCC.INP"`: The input-oriented BCC radial model. • `"DDF"`: The directional distance function. • `"RSL.OUT"`: The output-oriented Russell model. • `"RSL.INP"`: The input-oriented Russell model. • `"WAM.MIP"`: The weighted additive model with Measure of Inefficiency Proportion. • `"WAM.RAM"`: The weighted additive model with Range Adjusted Measure of Inefficiency.
`efficiencyJitter`	It returns a jitter plot (from `ggplot2`) that represents the dispersion of the efficiency scores of the set of DMUs in the leaf nodes of an Efficiency Analysis Trees model. Mean and standard deviation of scores are shown.
`efficiencyDensity`	It returns a density plot (from `ggplot2`) to compare the distribution of efficiency scores between two given models (`"EAT"`, `"FDH"`, `"CEAT"`, `"DEA"` and `"RFEAT"` are available).

Continued on next page

Table 1 – continued from previous page

Function	Description
`plotEAT`	It returns a plot of the tree-structure (from `ggparty` and `partykit`) of an Efficiency Analysis Trees model.
`frontier`	It returns a plot (from `ggplot2`) of the estimated production function obtained by an Efficiency Analysis Trees model in a two-dimensional scenario (1 input and 1 output). Optionally, the FDH frontier can be plotted.
`predict`	Generic function to predict the expected output by an `EAT` object. The result is a `data.frame` with the predicted values.
`rankingEAT`	It returns a `data.frame` with the scores of variable importance obtained by an Efficiency Analysis Trees model and optionally a barplot representing the variable importance.
Subsection 2: Random Forest for Efficiency Analysis Trees	
`RFEAT`	It generates a Random Forest for Efficiency Analysis Trees model and returns an `RFEAT` object.
`bestRFEAT`	It computes the root mean squared error (RMSE) for a set of Random Forest for Efficiency Analysis Trees models made up of a set of user-entered hyper-parameters. These models are fitted with a training sample and evaluated with a test sample.
`efficiencyRFEAT`	It computes the efficiency scores of a set of DMUs through a Random Forest for Efficiency Analysis Trees model and returns a `data.frame`. The FDH scores can also be computed. Only the output-oriented BCC radial model is available.
`plotRFEAT`	It returns a line plot (from `ggplot2`) with the Out-of-Bag (OOB) error for a random forest consisting of k trees.
`predict`	Generic function to predict the expected output by an `RFEAT` object. The result is a `data.frame` with the predicted values.
`rankingRFEAT`	It returns a `data.frame` with the scores of variable importance obtained by a Random Forest for Efficiency Analysis Trees model and optionally a barplot representing the variable importance.
Subsection 3: Convexified Efficiency Analysis Trees	
`efficiencyCEAT`	It computes the efficiency scores of a set of DMUs through a Convexified Efficiency Analysis Trees model and returns a `data.frame`. The DEA scores can also be computed. Alternative mathematical programming models for calculating the efficiency scores are: • `"BCC.OUT"`: The output-oriented BCC radial model. • `"BCC.INP"`: The input-oriented BCC radial model. • `"DDF"`: The directional distance function. • `"RSL.OUT"`: The output-oriented Russell model. • `"RSL.INP"`: The input-oriented Russell model. • `"WAM.MIP"`: The weighted additive model with Measure of Inefficiency Proportion. • `"WAM.RAM"`: The weighted additive model with Range Adjusted Measure of Inefficiency.
Subsection 4: Functions for data simulation	
`Y1.sim`	It returns a `data.frame` with simulated data in a single output scenario (1, 3, 6, 9, 12 and 15 inputs can be generated).

Continued on next page

Table 1 – continued from previous page

Function	Description
`X2Y2.sim`	It returns a `data.frame` with simulated data in a scenario with 2 inputs and 2 outputs.

2 Background

Efficiency Analysis Trees

In this section, we briefly introduce the main fundaments of Efficiency Analysis Trees. Nevertheless, we first need to introduce some notation related to the standard Free Disposal Hull (FDH) and Classification and Regression Trees (CART) techniques.

We consider the observation of n Decision Making Units (DMUs), which consumes $\mathbf{x}_i = (x_{1i}, ..., x_{mi}) \in R_+^m$ quantity of inputs for the production of $\mathbf{y}_i = (y_{1i}, ..., y_{si}) \in R_+^s$ quantity of outputs[1]. The dataset is denoted in a compact way as $\aleph = \{(\mathbf{x}, \mathbf{y})\}_{i=1,...,n}$. The so-called production possibility set or technology, which is the set of technically feasible combinations of $(\mathbf{x}, \mathbf{y})$, is defined, in general terms, as:

$$\psi = \left\{(\mathbf{x}, \mathbf{y}) \in R_+^{m+s} : \mathbf{x} \textit{ can produce } \mathbf{y}\right\}. \tag{1}$$

On this set, certain assumptions are usually made (see, Färe and Primont 1995), such as: monotonicity (free disposability) of inputs and outputs, which means that if $(\mathbf{x}, \mathbf{y}) \in \psi$, then $(\mathbf{x'}, \mathbf{y'}) \in \psi$, as long as $\mathbf{x'} \geq \mathbf{x}$ and $\mathbf{y'} \leq \mathbf{y}$; and convexity, i.e., if $(\mathbf{x}, \mathbf{y}) \in \psi$ and $(\mathbf{x'}, \mathbf{y'}) \in \psi$, then $\lambda (\mathbf{x}, \mathbf{y}) + (1 - \lambda) (\mathbf{x'}, \mathbf{y'}) \in \psi$, $\forall \lambda \in [0, 1]$. In the case of the FDH estimator, only free disposability is assumed. Additionally, FDH is assumed to be deterministic. In other words, the production possibility set determined by FDH always contains all the observations that belong to the data sample and, graphically, the production frontier envelops the data cloud from above. Also, FDH satisfies the minimal extrapolation postulate, which is associated with the typical problem-solving principle of Occam's razor. That is, additional requirements are needed to select the right estimator because there are a lot of possible estimators that can meet free disposability and the deterministic quality. In this sense, according to Occam's razor, the most conservative estimate of the production frontier would be that related to a surface that would envelop the data from above, satisfy free disposability and, at the same time, be as close as possible to the data cloud. In contrast, the DEA estimator requires stronger assumptions, such as convexity of the set ψ.

With regard to the measurement of technical efficiency, a certain part of the set ψ is actually of interest. It is the efficient frontier or production frontier of ψ, which is defined as $\partial(\boldsymbol{\psi}) := \{(\mathbf{x}, \mathbf{y}) \in \boldsymbol{\psi} : \hat{\mathbf{x}} < \mathbf{x}, \hat{\mathbf{y}} > \mathbf{y} \Rightarrow (\hat{\mathbf{x}}, \hat{\mathbf{y}}) \notin \boldsymbol{\psi}\}$. Technical efficiency is understood as the distance from a point belonging to ψ to the production frontier $\partial(\boldsymbol{\psi})$. In particular, Deprins and Simar (1984) proposed the FDH estimator of the set ψ from the dataset $\aleph$ as:

$$\hat{\boldsymbol{\psi}}_{FDH} = \left\{(\mathbf{x}, \mathbf{y}) \in R_+^{m+s} : \exists i = 1, ..., n \text{ such that } \mathbf{y} \leq \mathbf{y}_i, \mathbf{x} \geq \mathbf{x}_i\right\}. \tag{2}$$

The FDH technique is very attractive because it is based on very few suppositions, but it suffers from overfitting due to its construction. This problem is shared by other well-known data-based approaches. For example, Classification and Regression Trees (CART), a technique that belongs to the field of machine learning, suffer problems of overfitting when a deep tree is developed. However, this problem can be fixed using a cross-validation process to prune the deep tree. The principle behind CART is relatively simple: a certain criterion is chosen to recursively generate binary partitions of the data until a meaningful division is no longer possible or a stopping rule is maintained. The graphic result of this approach is a tree that starts at the root node, develops through the intermediate nodes and ends at the terminal nodes, also known as leaves. The binary nature of CART is represented by each parent node, except for the leaves, giving rise to two child nodes.

Next, we briefly introduce the recent technique named Efficiency Analysis Trees by Esteve et al. (2020). This technique allows the estimation of production frontiers, fulfilling the common axioms of microeconomics, through a data-based approach that is not founded on any particular distribution on the data noise and, in addition, creates a step function as a estimator. It shares these characteristics with the FDH technique, but the overfitting problem related to FDH can be solved through cross-validation based on pruning.

We now introduce the main steps of the algorithm linked to the Efficiency Analysis Trees technique.

[1]We use bold for denoting vectors, and non-bold for scalars.

Let us assume that we have a node t in the tree structure to be split. This node contents a subset of the original sample $\aleph$. The algorithm has to select an input variable j, $j = 1, ..., m$, and a threshold $s_j \in S_j$, where S_j is the set of possible thresholds for variable j, such that the sum of the mean squared error (MSE) calculated for the data that belong to the left child node t_L and the MSE corresponding to the data belonging to the right child node t_R is minimized. The data of the left child node t_L satisfies the condition $x_j < s_j$, while the data of the right child node t_R satisfies the condition $x_j \geq s_j$. Additionally, in the algorithm, the set S_j is defined from the observed values of the input j in the data sample $\aleph$. Formally, the split consists in selecting the combination $\left(x_j, s_j\right)$ which minimizes $R\left(t_L\right) + R\left(t_R\right) = \frac{1}{n}\sum_{(x_i,y_i)\in t_L}\sum_{r=1}^{s}\left(y_{ri} - y_r\left(t_L\right)\right)^2 + \frac{1}{n}\sum_{(x_i,y_i)\in t_R}\sum_{r=1}^{s}\left(y_{ri} - y_r\left(t_R\right)\right)^2$, where $y_r\left(t\right)$ denotes the estimation of the r-th output of the node t. One of the most important aspects in the production context is how to define $y_r\left(t\right)$ in each node for fulfilling the free disposability property during the growing process of the tree. In this sense, the notion of Pareto-dominance between nodes introduced in Esteve et al. (2020) is really relevant.

As described above, each node t is defined by a series of conditions in the input space as $\left\{x_j < s_j\right\}$ or $\left\{x_j \geq s_j\right\}$. In this sense, after executing the split, a region in the input space is created. This region in the input space is called the "support" of node t and is defined as $\text{supp}\left(t\right) = \left\{\mathbf{x} \in R_+^m : a_j^t \leq x_j < b_j^t, j = 1, ..., m\right\}$. The parameters a_j^t and b_j^t are originated from the several thresholds selected during the splitting process. Giving the notion of support of a node, it is possible to establish the concept of Pareto-dominance. Let $k = 1, ..., K$ be the total number of splits executed. Let $T_k\left(\aleph\right)$ be the tree built after the k-th split. Let $\tilde{T}_k\left(\aleph\right)$ be the set of leaves in the tree $T_k\left(\aleph\right)$. More notation: let $t^* \in \tilde{T}_k\left(\aleph\right)$ be the node to be split in a certain step of the algorithm, then $T\left(k|t^* \to t_L, t_R\right)$ denotes the tree associated with this specific split. Let $k = 1, ..., K$ and $t \in \tilde{T}_k\left(\aleph\right)$, then the set of Pareto-dominant nodes of node t is defined as $P_{T_k(\aleph)}\left(t\right) = \{t' \in \tilde{T}_k\left(\aleph\right) - t : \exists \mathbf{x} \in \text{supp}\left(t\right), \exists \mathbf{x'} \in \text{supp}\left(t'\right) \text{ such that } \mathbf{x'} \leq \mathbf{x}\}$. $P_{T_k(\aleph)}\left(t\right)$ contains all the nodes such that at least one input vector in its corresponding support, non-necessarily observed, dominates at least one input vector belonging to the support of node t (in the Pareto sense). To do so, in practice, it is only necessary to compare the components of $\mathbf{a}^{t'}$ and $\mathbf{b}^t$. Specifically, $\mathbf{a}^{t'} < \mathbf{b}^t$ if and only if $t' \in P_{T_k(\aleph)}\left(t\right)$.

Now, we return to how to estimate the outputs in each child node with the aim of guaranteeing the satisfaction of free disposability. For any node $t^* \in \tilde{T}_k\left(\aleph\right)$, the way to estimate the value of the outputs for the right child node is through the estimation of the outputs of its parent node, i.e., $y_r\left(t_R\right) = y_r(t^*),\ r = 1, ..., s$, while the estimation of outputs for the left child node is:

$$y_r\left(t_L\right) = \max\left\{\max\left\{y_{ri} : \left(\mathbf{x}_i, \mathbf{y}_i\right) \in t_L\right\}, y_r\left(I_{T(k|t^*\to t_L,t_R)}\left(t_L\right)\right)\right\},\ r = 1, ..., s, \tag{3}$$

where $y_r\left(I_{T(k|t^*\to t_L,t_R)}\left(t_L\right)\right) = \max\left\{y_r\left(t'\right) : t' \in I_{T(k|t^*\to t_L,t_R)}\left(t_L\right)\right\}$ and $y_r\left(t'\right)$ is the estimation of the output y_r at node $t' \in \tilde{T}\left(k|t^* \to t_L, t_R\right)$, $r = 1, ..., s$. This way of estimating the output values guarantees free disposability.

Accordingly, the algorithm selects the best pair $\left(x_{j^*}, s_{j^*}\right)$ such that the sum of the MSE of the left and right child nodes is minimized. Once the split of node t^* is executed, the tree $T\left(k|t^* \to t_L^*, t_R^*\right)$ is obtained. This process continues until bipartition is not possible because all the data in a node have the same input values or a certain stopping rule is satisfied. The usual stopping rule is $n\left(t\right) \leq n_{\min} = 5$, where $n(t)$ is the sample size of node t. The final tree built is denoted as $T_{\max}\left(\aleph\right)$, which usually is a deep tree.

$T_{\max}\left(\aleph\right)$ suffers from the same problem as FDH, i.e., overfitting. Esteve et al. (2020) proposed to prune the tree exploiting the same technique as Breiman et al. (1984). This pruning process resorts to the notion of the error-complexity measure $R_\alpha\left(T\left(\aleph\right)\right)$, which is a combination between a measure of the accuracy of the tree, defined as the sum of the MSE determined at each leaf node, and a measure of the number of leaf nodes. Also, $R_\alpha\left(T\left(\aleph\right)\right)$ depends on a parameter α, which compensates the values of the two components of the error: $R_\alpha\left(T\left(\aleph\right)\right) = R\left(T\left(\aleph\right)\right) + \alpha\left|\tilde{T}\left(\aleph\right)\right|$. The idea behind the pruning of $T_{\max}\left(\aleph\right)$ is to minimize $R_\alpha\left(T\left(\aleph\right)\right)$. The pruning process is also based on cross-validation (see Breiman et al. (1984) for more details). The tree resulting from the pruned process is $T^*\left(\aleph\right)$. This tree doesn't suffer from the overfitting problem. For this reason, the use of $T^*\left(\aleph\right)$ is recommended rather than $T_{\max}\left(\aleph\right)$, unless a descriptive analysis of the sample is required.

Finally, $\mathbf{d}_{T^*(\aleph)}\left(\mathbf{x}\right)$ will denote hereinafter the multi-dimensional estimator defined from $T^*\left(\aleph\right)$ and the sample $\aleph$, i.e., $d_{rT^*(\aleph)}\left(\mathbf{x}\right) = \sum_{t\in T^*(\aleph)} y_r\left(t\right) I\left(\mathbf{x} \in t\right)$, for all $r = 1, ..., s$, with $I\left(\cdot\right)$ being the indication function. From this estimator, it is possible to define a production possibility set or technology estimated from the Efficiency Analysis Trees technique as:

$$\hat{\Psi}_{T^*(\aleph)} = \left\{\left(\mathbf{x}, \mathbf{y}\right) \in R_+^{m+s} : \mathbf{y} \leq \mathbf{d}_{T^*(\aleph)}\left(\mathbf{x}\right)\right\}. \tag{4}$$

$\hat{\Psi}_{T^*(\aleph)}$ satisfies free disposability and the deterministic quality.

By analogy with the existing relationship between FDH and DEA, it is possible to derive an estimation of Ψ by the convexification of the set $\hat{\Psi}_{T^*}$. In this sense, the convexification of the production possibility set derived from EAT would be as follows:

$$conv\left(\hat{\Psi}_{T^*}\right) = \left\{(\mathbf{x},\mathbf{y}) \in R_+^{m+s} : \mathbf{x} \geq \sum_{t\in\tilde{T}^*} \lambda_t \mathbf{a}^t, \mathbf{y} \leq \sum_{t\in\tilde{T}^*} \lambda_t \mathbf{d}_{T^*}\left(\mathbf{a}^t\right), \sum_{t\in\tilde{T}^*} \lambda_t = 1, \boldsymbol{\lambda} \geq 0_{|\tilde{T}^*|}\right\}. \quad (5)$$

Under the convexity assumption, the EAT methodology is known as the Convexified Efficiency Analysis Trees technique (hereinafter referred to as CEAT) (see Aparicio et al. 2021).

Random Forest for Efficiency Analysis Trees

In this section, we briefly describe the extension of the approach by Esteve et al. (2020) to the context of using ensembles of trees to provide estimates of production frontiers (see Esteve et al. 2021). Specifically, we briefly revise the way to adapt the standard Random Forest (Breiman 2001) for estimating production frontiers satisfying fundamental postulates of microeconomics, such as free disposability. The adaptation of Random Forest to the estimation of production frontiers by Esteve et al. (2021) is the first one that focuses on the introduction of a methodology for measuring technical efficiency that is robust to the resampling of data and, at the same time, to the specification of input variables.

Data robustness and resampling methods for input modeling are both topics of interest in the literature on technical efficiency measurement. Regarding robustness to data, Simar and Wilson (1998, 2000; 2000) were the first ones to adapt the bootstrapping methodology (Efron 1979) to the context of DEA and FDH. As regards the importance of the robustness of input and output variables in non-parametric efficiency analysis, since the beginning of DEA and FDH, researchers have always been aware that the selection of input and output variables to be considered in efficiency analysis is one of the crucial issues in the specification of the model. In practice, the researchers' previous experience may lead to the selection of some inputs and outputs considered essential to represent the underlying technology. However, there may be other variables whose inclusion in the model the analyst is not always sure of (Pastor, Ruiz, and Sirvent 2002). Some approaches focus on balancing the experience of researchers with the information provided by observations (see, for example, Banker 1993, 1996; Pastor, Ruiz, and Sirvent 2002). Another recent approach is based, in contrast, on determining efficiency scores that are robust to variable selection by considering all the possible combinations of inputs and outputs and their aggregation (Landete, Monge, and Ruiz 2017).

On the whole, Random Forest (Breiman 2001) is an ensemble learning method that works by constructing a multitude of decision trees by CART (Breiman et al. 1984) at training time and aggregating the information of the individual trees in a final prediction value. In particular, when Random Forest is applied to regression problems, the final estimator corresponds to the mean of each individual prediction (Breiman 2001). Random Forest modifies the growing process of an individual tree as follows, by: (i) applying bootstrapping on the data training for each individual tree and (ii) selecting a random subset of the predictors in each iteration. In this way, given a learning sample $\aleph$ of size n, Random Forest repeatedly selects random samples of size n with replacement of the set $\aleph$. Then, the method fits the trees to these samples but, to do this, it uses a modified tree learning algorithm that chooses, in each candidate division of the learning process, a random subset of predictors. The reason for doing this is due to the instability of the model. It is known that individual decision trees, such as CART, are very unstable (Berk 2016). This means that completely different tree structures are given when the training data is modified slightly. In this way, the result of applying Random Forest is an estimator that overcomes overfitting and instability problems in general, resulting in a substantial reduction in variance.

The algorithm associated with the adaptation of the Random Forest technique to the world of technical efficiency assessment, called RF+EAT, is introduced in Esteve et al. (2021). The steps that must be carried out in Random Forest for Efficiency Analysis Trees are shown in Algorithm 1. This algorithm is based on the typical algorithm of Random Forest that can be found in Kuhn, Johnson, et al. (2013). In Algorithm 1, the first step consists of selecting the number of trees that will make up the forest, that is, the hyperparameter p. Then, p (bootstrap) random samples from the original data sample with replacement are generated. Next, the Efficiency Analysis Trees algorithm by Esteve et al. (2020) is applied to each subsample applying the stopping rule $n(t) \leq n_{\min}$, but without pruning. Also, in this algorithm, $n_{\min}$ is treated as an additional hyperparameter that could be tuned. During the execution of the Efficiency Analysis Trees algorithm, a subset of input variables (*mtry*) from the original set is randomly selected each time the splitting subroutine is applied. To do that, one of the

following five thumb rules is used following the literature:

- Breiman's Rule: $mtry = \frac{m}{3}$,
- Rule DEA1: $mtry = \frac{n(t)}{2} - s$ (Golany and Roll 1989; Homburg 2001),
- Rule DEA2: $mtry = \frac{n(t)}{3} - s$ (Nunamaker 1985; Banker et al. 1989; Friedman and Sinuany Stern 1998; Raab and Lichty 2002),
- Rule DEA3: $mtry = \frac{n(t)}{2s}$ (Dyson et al. 2001),
- Rule DEA4: $mtry = \min\left\{\frac{n(t)}{s}, \frac{n(t)}{3} - s\right\}$ (W. Cooper et al. 2007).

Input: p, number of trees
$\aleph$, original data
Output: $\{T(\aleph_q) : q = 1, ..., p\}$
for $q = 1$ ***to*** p **do**
  $\aleph_q$:= Bootstrap sample of $\aleph$
  $T(\aleph_q)$:= Efficiency Analysis Tree trained on $\aleph_q$
  foreach *split* **do**
    Randomly in $T(\aleph_q)$ selects $mtry(\leq m)$ of the original inputs using a specific rule;
    Select the best input in $T(\aleph_q)$ among the *mtry* inputs and split the data
  end
  $T(\aleph_q)$ is completed when $n(t) \leq n_{min}, \forall t$ leaf node of $T(\aleph_q)$
  ($T(\aleph_q)$ is not pruned)
end

Algorithm 1: Random Forest for Efficiency Analysis Trees algorithm for estimating production frontiers

Once Algorithm 1 has been applied, p fitted trees are determined with the aim of obtaining an output estimation giving an input vector $\mathbf{x} \in R_+^m$. In this regard, we have $T(\aleph_1), ..., T(\aleph_p)$ tree structures derived from the application of the Efficiency Analysis Trees algorithm on the p bootstrap subsamples $\aleph_1, ..., \aleph_p$. Given an input vector $\mathbf{x} \in R_+^m$, an output estimator is determined by averaging the individual estimator corresponding to each tree:

$$\mathbf{y}^{RF+EAT(\aleph)}(\mathbf{x}) := \frac{1}{p}\sum_{q=1}^{p}\mathbf{d}_{T(\aleph_q)}(\mathbf{x}). \tag{6}$$

where $\mathbf{d}_{T(\aleph_q)}(\mathbf{x})$ denotes the output estimator associated with each tree structure $T(\aleph_q)$, given an input vector $\mathbf{x} \in R_+^m$.

In addition, this estimator allows the technology or production possibility set to be defined as:

$$\hat{\Psi}_{RF+EAT} = \left\{(\mathbf{x}, \mathbf{y}) \in R_+^{m+s} : \mathbf{y} \leq \mathbf{y}^{RF+EAT(\aleph)}(\mathbf{x})\right\}. \tag{7}$$

As happens with the standard Random Forest, Random Forest for Efficiency Analysis Trees also exploits the Out-Of-Bag (OOB) concept. The OOB estimate at observation $(\mathbf{x}_i, \mathbf{y}_i)$ consists in evaluating the prediction of the ensemble just using the individual models $T(\aleph_q)$ whose corresponding bootstrap samples $\aleph_q$ are such that $(\mathbf{x}_i, \mathbf{y}_i) \notin \aleph_q$. From this definition, the generalization error is defined as the average of the OOB estimates calculated over all the observations in the learning sample $\aleph$:

$$err^{RF+EAT(\aleph)} = \frac{1}{n}\sum_{(x_i,y_i)\in\aleph}\sum_{r=1}^{s}\left(y_{ri} - y_r^{RF+EAT(\aleph)}(\mathbf{x}_i)\right)^2. \tag{8}$$

The generalization error is useful for determining a measure of variable importance, which can be used for creating a sorted list of inputs $x_1, ..., x_m$. The way to calculate the input importance of variable x_j is: firstly, generate a new database, $\aleph^j$, identical to the original one $\aleph$, where specifically the values of variable x_j were randomly permuted; secondly, apply Algorithm 1 on the new 'virtual' learning sample $\aleph^j$; thirdly, determine the value of the generalization error, i.e., $err^{RF+EAT(\aleph^j)}$; and, finally, calculate the percentage increase of the generalization error when variable x_j is shuffled as:

$$\%Inc^{RF+EAT}\left(x_j\right) = 100 \cdot \left(\frac{err^{RF+EAT\left(\aleph^j\right)} - err^{RF+EAT(\aleph)}}{err^{RF+EAT(\aleph)}}\right). \tag{9}$$

3 Data structure

Data are managed as a regular R `data.frame` in the **eat** functions (`matrix` is often accepted but will be converted to a `data.frame` in the functions pre-processing). The main functions of the **eat** package are `EAT()` and `RFEAT()`, which return structured objects named `EAT` and `RFEAT`, respectively. These objects contain fields with relevant information such as the estimation results or the arguments introduced by the user in the function call.

The fields of the `EAT` object are the following:

- `data`: Contains the input and output variables.
  - `df`: Data introduced by the user in a `data.frame` structure after being preprocessed.
  - `x`: Input indexes in `df`.
  - `y`: Output indexes in `df`.
  - `input_names`: Name of the input variables in `df`.
  - `output_names`: Name of the output variables in `df`.
  - `row_names`: Name of the observations in `df`.
- `control`: Contains the hyperparameters selected by the user.
  - `fold`: Number of folds in which is divided `df` to apply cross-validation.
  - `numStop`: Minimum number of observations in a node.
  - `max.leaves`: Maximum number of leaf nodes.
  - `max.depth`: Maximum number of nodes between the root node (not included) and the furthest leaf node.
  - `na.rm`: A logical variable that indicates if `NA` rows should be ignored.
- `tree`: `list` containing the nodes of the fitted Efficiency Analysis Trees model. Each node is made up of the following elements:
  - `id`: Node index
  - `F`: Father node index.
  - `SL`: Left child node index.
  - `SR`: Right child node index.
  - `index`: Set of indexes corresponding to the observations in a node.
  - `R`: Error at the node.
  - `xi`: Index of the variable that produces the split in a node.
  - `s`: Threshold of the variable `xi`.
  - `a`: The components of the vector $\mathbf{a}^t$.
  - `b`: The components of the vector $\mathbf{b}^t$.
- `nodes_df`: Contains the following information related to the nodes of the fitted Efficiency Analysis Trees model in a `data.frame` structure:
  - `id`: Node index
  - `N`: Number of observations in a node.
  - `Proportion`: Proportion of observations in a node.
  - `y`: Fitted values.
  - `R`: Error at the node.
  - `index`: Indexes of the observations in a node.
- `model`: Contains the following information related to the fitted Efficiency Analysis Trees model:
  - `nodes`: Number of nodes in the tree.
  - `leaf_nodes`: Number of leaf nodes in the tree.
  - `a`: The components of the vector $\mathbf{a}^t$.
  - `y`: Output estimation for each leaf node.

Regarding the `RFEAT` object, it contains the following fields:

- `data`: same fields as the `EAT` object.
- `control`: Contains the hyperparameters selected by the user.

  - numStop: Minimum number of observations in a node.
  - m: Number of trees that make up the random forest.
  - s_mtry: Number of inputs that can be randomly selected in each split.
  - na.rm: A logical variable that indicates if NA rows should be ignored.
- forest: A list containing the individual Efficiency Analysis Trees that make up the random forest.
- Error: The Out-of-Bag error at the random forest.
- OOB: A list containing the observations used for training each Efficiency Analysis Tree that makes up the random forest.

Dataset and statistical sources

```
# We load the library
library("eat")

# We load the data
data("PISAindex")
```

We illustrate all the models presented in this paper resorting to a single dataset (PISAindex) available in the **eat** package. Our dataset consists of 72 countries with 3 outputs and 13 inputs. The output data have been collected by the PISA (Programme for International Student Assessment) 2018 survey (OECD 2018) and refers to the average score in mathematics, reading and science domains for schools in these countries. Regarding the input data, the variables have been collected from the Social Progress Index (2018) and are related to the socioeconomic environment of these countries. These inputs can be classified into four blocks as follows:

- Basic Human Needs:
  - Nutrition and Basic Medical Care (NBMC)
  - Water and Sanitation (WS)
  - Shelter (S)
  - Personal Safety (PS).
- Foundations of Wellbeing:
  - Access to Basic Knowledge (ABK)
  - Access to Information and Communications (AIC)
  - Health and Wellness (HW)
  - Environmental Quality (EQ).
- Opportunity:
  - Personal Rights (PR)
  - Personal Freedom and Choice (PFC)
  - Inclusiveness (I)
  - Access to Advanced Education (AAE).
- Economy:
  - Gross Domestic Product based on Purchasing Power Parity (GDP_PPP).

Finally, in order to simplify the examples and reduce computation time, a subset of variables is selected as follows:

```
# Inputs (5): PR, PFC, I, AAE, GDP_PPP
# Outputs (3): S_PISA, R_PISA, M_PISA
PISAindex <- PISAindex[, c(3, 4, 5, 14, 15, 16, 17, 18)]

head(PISAindex)
```

```
#>     S_PISA R_PISA M_PISA    PR   PFC     I   AAE GDP_PPP
#> SGP    551    549    569 71.70 87.90 48.26 74.31  97.745
#> JPN    529    504    527 94.07 82.40 62.32 81.29  41.074
#> KOR    519    514    526 92.71 79.06 63.54 86.32  41.894
#> EST    530    523    523 95.67 84.10 55.58 73.16  35.308
#> NLD    503    485    519 96.34 89.04 75.82 82.99  56.455
#> POL    511    512    516 86.41 78.25 57.58 76.21  31.766
```

Table 2 reports the descriptive statistics for these variables (outputs and inputs).

Table 2: Descriptive statistics (averages, standard deviations, minimum, median and maximum) of input–output.

variable	type	mean	sd	min	median	max
S_PISA	output	455.06	48.32	336.00	466.00	551.00
R_PISA	output	450.89	50.52	340.00	466.00	549.00
M_PISA	output	454.81	52.17	325.00	463.50	569.00
PR	input	81.62	17.98	21.14	88.40	98.07
PFC	input	75.42	11.03	47.25	78.19	91.65
I	input	54.17	17.07	12.37	55.51	81.91
AAE	input	69.87	10.75	48.37	71.65	90.43
GDP_PPP	input	36.04	21.91	7.44	31.42	114.11

4 Basic functions of the library

In this section, we introduce the main functions of the library related to Efficiency Analysis Trees and Random Forest for Efficiency Analysis Trees. To execute the following examples, the package **eat** must be loaded and the seed 100 must be set for reproducibility.

```
# We set the seed
set.seed(100)
```

The EAT basic model

The basic model of Efficiency Analysis Trees that we explained in subsection Efficiency Analysis Trees can be implemented in R using the function `EAT()`:

```
EAT(
  data, x, y,
  numStop = 5,
  fold = 5,
  max.depth = NULL,
  max.leaves = NULL,
  na.rm = TRUE
)
```

The `EAT()` function is the cornerstone of the **eat** library. The minimum arguments of this function are the data (`data`) containing the study variables, the indexes of the predictor variables or inputs (`x`) and the indexes of the predicted variables or outputs (`y`). Additionally, the `numStop`, `fold`, `max.depth` and `max.leaves` arguments are included for more experienced users in the fields of machine learning and tree-based models. Modifying these four hyperparameters allows obtaining different frontier estimates and therefore selecting the one that best suits the needs of the analysis. The description of these parameters is as follows:

- `numStop` refers to the minimum number of observations in a node to be split and is directly related to the size of the tree. The higher the value of `numStop`, the smaller the size of the tree.
- `fold` refers to the number of parts in which the `data` is divided to apply the cross-validation technique. Variations in the `fold` argument are not directly related to the size of the tree.
- `max.depth` limits the number of nodes between the root node (not included) and the furthest leaf node. When this argument is introduced, the typical process of growth-pruning is not carried out. In this case, the tree is allowed to grow to the required depth.
- `max.leaves` determines the maximum number of leaf nodes. As in `max.depth`, the process of growth-pruning is not performed. In this respect, the tree grows until the required number of leaf nodes is reached, and then, the tree is returned.

Notice that including the arguments `max.depth` or `max.leaves` reduces the computation time by eliminating the pruning procedure. However, the pruning process is preferred if the objective of the study is inferential instead of simply descriptive. If both are included at the same time, a `warning` message is displayed and only `max.depth` is used.

As an example, using data from subsection Dataset and statistical sources, we next create a multi response tree using the suitable code as follows. Results are returned as an `EAT` object, as explained in Section Data structure.

```
modelEAT <- EAT(data = PISAindex, x = 4:8, y = 1:3)
```

The RFEAT basic model

The basic model of Random Forest for Efficiency Analysis Trees that we explained in subsection Random Forest for Efficiency Analysis Trees can be implemented in R using the function `RFEAT()`:

```
RFEAT(
  data, x, y,
  numStop = 5,
  m = 50,
  s_mtry = "BRM",
  na.rm = TRUE
)
```

The `RFEAT()` function has also been developed with the aim of providing greater statistical robustness to the results obtained by the `EAT()` function. The `RFEAT()` function requires the `data` containing the variables for the analysis, `x` and `y` corresponding to the inputs and outputs indexes respectively, the minimum number of observations in a node for a split to be attempted (`numStop`) and `na.rm` to ignore observations with `NA` cells. All these arguments are used for the construction of the p (this is denoted with `m` in the `RFEAT()` function) individual Efficiency Analysis Trees that make up the random forest. Finally, the argument `s_mtry` indicates the number of inputs that can be randomly selected in each split. It can be set as any integer although there are also certain predefined values. Let m be the number of inputs, let s be the number of outputs and let $n(t)$ be the number of observations in a node. Then, the predefined values for `s_mtry` are:

- `BRM` = $\frac{m}{3}$,
- `DEA1` = $\frac{n(t)}{2} - s$,
- `DEA2` = $\frac{n(t)}{3} - s$,
- `DEA3` = $\frac{n(t)}{2s}$,
- `DEA4` = $\min\left\{\frac{n(t)}{s}, \frac{n(t)}{3} - s\right\}$.

As an example, using data from subsection Dataset and statistical sources, we next create a forest with 30 trees. Results are returned as an `RFEAT` object, as explained in Section Data structure.

```
modelRFEAT <- RFEAT(data = PISAindex, x = 4:8, y = 1:3, m = 30)
```

Predictions

The estimators of the Efficiency Analysis Trees and Random Forest for Efficiency Analysis Trees can be computed in R using the function `predict()`:

```
predict(
  object,
  newdata,
  x, ...
)
```

Regarding the arguments of `predict()`, `object` can be an `EAT` or an `RFEAT` object, `newdata` refers to a `data.frame` and `x` to the set of inputs to be used. This function returns a `data.frame` with the expected output for a set of observations. For predictions using an `EAT` object, only one tree is used. However, for the RFEAT model, the output is predicted by each of the p (`m` in the `RFEAT()` function) individual trees trained and subsequently the mean value of all predictions is obtained.

As an example, we next evaluate the last 3 DMUs from the data of subsection Dataset and statistical sources and the corresponding EAT and RFEAT models. Results are returned in a `data.frame` structure with the output predictions:

```
predict(object = modelEAT, newdata = tail(PISAindex, 3), x = 4:8)
```

```
#>   S_PISA_pred R_PISA_pred M_PISA_pred
#> 1         428         424         437
#> 2         377         359         368
#> 3         377         359         368
```

```
predict(object = modelRFEAT, newdata = tail(PISAindex, 3), x = 4:8)

#>   S_PISA_pred R_PISA_pred M_PISA_pred
#> 1    439.9667    435.1333    441.2000
#> 2    402.0667    389.0667    403.9000
#> 3    399.0333    389.3333    399.6333
```

In the same way, the user can also create a new `data.frame` and calculate predictions for it as follows:

```
new <- data.frame(AAE = c(61, 72), PR = c(76, 81), I = c(41, 55), GDP_PPP = c(19, 31),
                  PFC = c(67, 78))

predict(object = modelEAT, newdata = new, x = 1:5)

#>   S_PISA_pred R_PISA_pred M_PISA_pred
#> 1         428         424         421
#> 2         481         479         488
```

Importance of predictor variables

The way to compute in R the predictor variables importance in the Efficiency Analysis Trees methodology is using the functions `rankingEAT()` or `rankingRFEAT()`:

```
# Through Efficiency Analysis Trees
rankingEAT(
  object,
  barplot = TRUE,
  threshold = 70,
  digits = 2
)

# Through Random Forest for Efficiency Analysis Trees
rankingRFEAT(
  object,
  barplot = TRUE,
  digits = 2
)
```

These functions allow a selection of variables by calculating a score of importance through Efficiency Analysis Trees or Random Forest for Efficiency Analysis Trees, respectively. These importance scores represent how influential each variable is in the model. Regarding the Efficiency Analysis Trees [`RankingEAT()`], the notion of surrogate splits by Breiman et al. (1984) was implemented. In this regard, the measure of importance of a variable x_j is defined as the sum over all nodes of the decrease in mean squared error produced by the best surrogate split on x_j at each node (see Definition 5.9 in Breiman et al. (1984)). Since only the relative magnitudes of these measures are interesting for researchers, the actual measures of importance that we report are normalized. In this way, the most important variable has always a value of 100, and the others are in the range 0 to 100. As for the Random Forest for Efficiency Analysis Trees [`RankingRFEAT()`], (9) was implemented for each input variable. Regarding the available arguments of the functions, the user can specify the number of decimal units (`digits`) and include a barplot (from `ggplot2`) with the scores of importance (`barplot`). Additionally, the `rankingEAT()` function allows to display a horizontal line in the graph to facilitate the cut-off point between important and irrelevant variables (`threshold`).

As an example, we next use the objects `modelEAT` (an `EAT` object from the `EAT()` function) and `modelRFEAT` (an `RFEAT` object from the `RFEAT()` function) created in the previous section to assess the predictors used. These functions return the name of the predictor variables, the scores of importance (in the range 0-100 for the `rankingEAT()` function) and a barplot (without horizontal line for the `rankingRFEAT()` function).

```
rankingEAT(object = modelEAT)

#> $scores
```

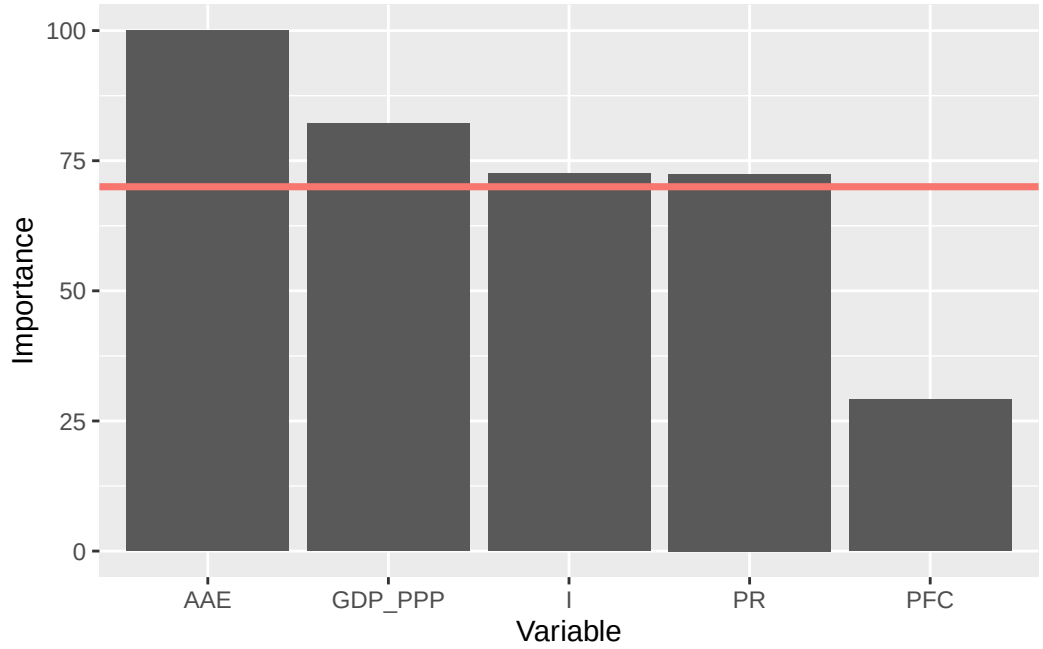

Figure 1: Barplot generated by applying 'rankingEAT()' to the PISAindex database to determine the ranking of variable importance.

```
#>         Importance
#> AAE         100.00
#> GDP_PPP      82.13
#> I            72.58
#> PR           72.45
#> PFC          29.07
#>
#> $barplot
```

```
rankingRFEAT(object = modelRFEAT)
```

```
#> $scores
#>         Importance
#> PR            1.75
#> PFC          -1.64
#> GDP_PPP      -2.16
#> I            -2.87
#> AAE          -3.67
#>
#> $barplot
```

Note that negative scores may appear when calculating the importance of variables using the `rankingRFEAT()` function. The appearance of this type of (negative) score can be understood as, if that variable were removed from the model, *ceteris paribus*, then an improvement in the predictive capacity of the model would be produced.

5 Basic EAT and RFEAT models

Efficiency scores are numerical values that indicate the degree of efficiency of a set of Decision Making Units (DMU). In the **eat** package, these scores can be calculated through an Efficiency Analysis Trees model, a Random Forest for Efficiency Analysis Trees model or a Convexified Efficiency Analysis Trees model. The code is as follows:

```
# For Efficiency Analysis Trees
```

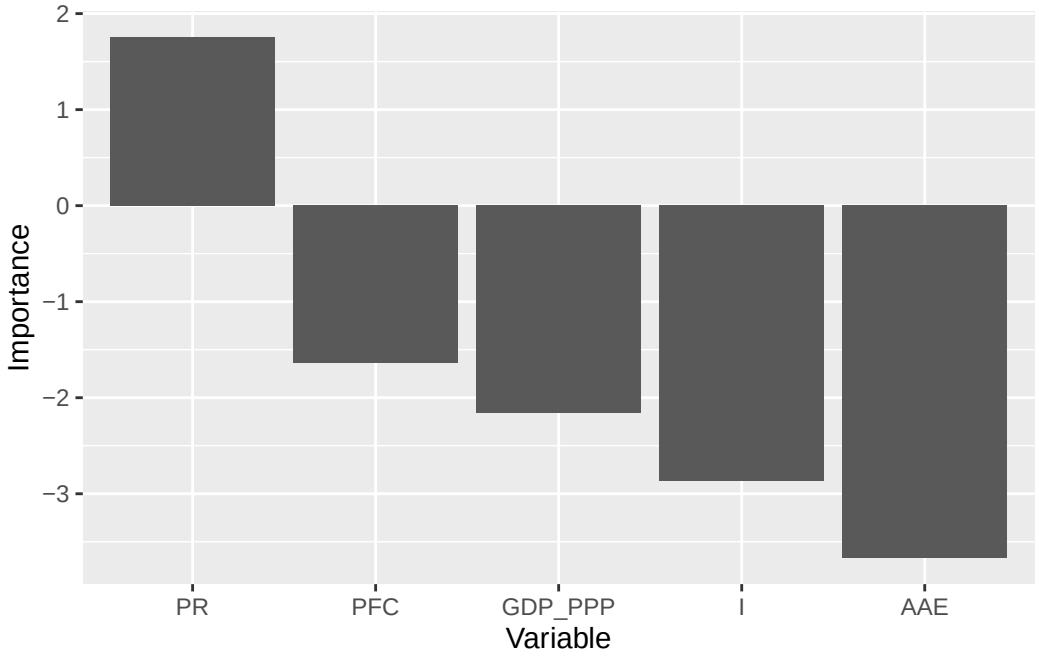

Figure 2: Barplot generated by applying 'rankingRFEAT()' to the PISAindex database to determine the ranking of variable importance.

```
efficiencyEAT(
  data, x, y, object, scores_model, digits = 3,
  FDH = TRUE, print.table = FALSE, na.rm = TRUE
)

# For Random Forest for Efficiency Analysis Trees
efficiencyRFEAT(
  data, x, y, object, digits = 3,
  FDH = TRUE, print.table = FALSE, na.rm = TRUE
)

# For Convexified Efficiency Analysis Trees
efficiencyCEAT(
  data, x, y, object, scores_model, digits = 3,
  DEA = TRUE, print.table = FALSE, na.rm = TRUE
)
```

A dataset (`data`) and the corresponding indexes of input(s) (`x`) and output(s) (`y`) must be entered. It is recommended that the `data` with the DMUs whose efficiency is to be calculated coincide with those used to estimate the frontier. However, it is also possible to calculate the efficiency scores for new `data`. The efficiency scores are calculated using the mathematical programming model included in the argument `scores_model`. The following models are available:

- `BCC.OUT`: The output-oriented radial model (Banker, Charnes, and Cooper 1984).
- `BCC.INP`: The input-oriented radial model (Banker, Charnes, and Cooper 1984).
- `RSL.OUT`: The output-oriented Russell model (Färe and Lovell 1978).
- `RSL.INP`: The input-oriented Russell model (Färe and Lovell 1978).
- `DDF`: The Directional Distance Function (Chambers, Chung, and Färe 1998).
- `WAM.MIP`: The Measure of Inefficiency Proportions as a type of Weighted Additive Model (Lovell and Pastor 1995).
- `WAM.RAM`: The Range-Adjusted Measure of Inefficiency as a type of Weighted Additive Model (Lovell and Pastor 1995; W. W. Cooper, Park, and Pastor 1999).

FDH or DEA scores can optionally be computed by setting `FDH = TRUE` or `DEA = TRUE`, respectively. Finally, a summary descriptive table of the efficiency scores can be displayed with the argument `print.table = TRUE`.

The output-oriented radial model

The output-oriented radial model determines the efficiency score for $(\mathbf{x}_k, \mathbf{y}_k) \in R_+^{m+s}$ by equiproportionally increasing all its outputs while maintaining inputs constant: $\phi(\mathbf{x}_k, \mathbf{y}_k) = \max\{\phi_k \in R : (\mathbf{x}_k, \phi_k \mathbf{y}_k) \in \Psi\}$.

The efficiency score $\phi(\mathbf{x}_k, \mathbf{y}_k)$ can be estimated through FDH by plugging $\hat{\Psi}_{FDH}$ from (2) into $\max\{\phi_k \in R : (\mathbf{x}_k, \phi_k \mathbf{y}_k) \in \Psi\}$ in place of Ψ. In that case, the optimization problem can be rewritten as a mixed-integer linear optimization program, as follows:

$$\begin{aligned} \phi^{FDH}(\mathbf{x}_k, \mathbf{y}_k) = \quad & \max \quad \phi, \\ & s.t. \\ & \quad \textstyle\sum_{i=1}^{n} \lambda_i x_{ji} \leq x_{jk}, \qquad j = 1, ..., m \\ & \quad \textstyle\sum_{i=1}^{n} \lambda_i y_{ri} \geq \phi y_{rk}, \quad r = 1, ..., s \\ & \quad \textstyle\sum_{i=1}^{n} \lambda_i = 1, \\ & \quad \lambda_i \in \{0, 1\}, \qquad i = 1, ..., n \end{aligned} \tag{10}$$

The Linear Programming model that should be solved under Data Envelopment Analysis would be:

$$\begin{aligned} \phi^{DEA}(\mathbf{x}_k, \mathbf{y}_k) = \quad & \max \quad \phi, \\ & s.t. \\ & \quad \textstyle\sum_{i=1}^{n} \lambda_i x_{ji} \leq x_{jk}, \qquad j = 1, ..., m \\ & \quad \textstyle\sum_{i=1}^{n} \lambda_i y_{ri} \geq \phi y_{rk}, \quad r = 1, ..., s \\ & \quad \textstyle\sum_{i=1}^{n} \lambda_i = 1, \\ & \quad \lambda_i \geq 0, \qquad i = 1, ..., n \end{aligned} \tag{11}$$

The following Mixed-Integer Linear Program should be solved for Efficiency Analysis Trees:

$$\begin{aligned} \phi^{EAT}(\mathbf{x}_k, \mathbf{y}_k) = \quad & \max \quad \phi, \\ & s.t. \\ & \quad \textstyle\sum_{t \in \tilde{T}^*} \lambda_t a_j^t \leq x_{jk}, \qquad j = 1, ..., m \\ & \quad \textstyle\sum_{t \in \tilde{T}^*} \lambda_t d_{rT^*}(\mathbf{a}^t) \geq \phi y_{rk}, \quad r = 1, ..., s \\ & \quad \textstyle\sum_{t \in \tilde{T}^*} \lambda_t = 1, \\ & \quad \lambda_t \in \{0, 1\}, \qquad t \in \tilde{T}^* \end{aligned} \tag{12}$$

In R, this model can be computed by setting `scores_model = "BCC.OUT"` in `efficiencyEAT()`:

```
scores <- efficiencyEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                        scores_model = "BCC.OUT", digits = 2,
                        print.table = TRUE)

#>  Model Mean Std. Dev. Min Q1 Median   Q3  Max
#>    EAT 1.03      0.04   1  1   1.01 1.01 1.16
#>    FDH 1.01      0.02   1  1   1.00 1.00 1.12

scores %>% sample_n(3)

#>     EAT_BCC_OUT FDH_BCC_OUT
#> CHL        1.09        1.05
#> SAU        1.05        1.00
#> CAN        1.01        1.01
```

Finally, the optimization model that should be solved for Convexified Efficiency Analysis Trees is:

$$\begin{aligned} \phi^{CEAT}(\mathbf{x}_k, \mathbf{y}_k) = \quad & \max \quad \phi, \\ & s.t. \\ & \quad \textstyle\sum_{t \in \tilde{T}^*} \lambda_t a_j^t \leq x_{jk}, \qquad j = 1, ..., m \\ & \quad \textstyle\sum_{t \in \tilde{T}^*} \lambda_t d_{rT^*}(\mathbf{a}^t) \geq \phi y_{rk}, \quad r = 1, ..., s \\ & \quad \textstyle\sum_{t \in \tilde{T}^*} \lambda_t = 1, \\ & \quad \lambda_t \geq 0, \qquad t \in \tilde{T}^* \end{aligned} \tag{13}$$

In R, this model can be computed by setting `scores_model = "BCC.OUT"` in `efficiencyCEAT()`:

```
scores <- efficiencyCEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
```

```
                        scores_model = "BCC.OUT", digits = 2,
                        print.table = TRUE)

#>  Model Mean Std. Dev. Min   Q1 Median   Q3  Max
#>   CEAT 1.11      0.07   1 1.05   1.09 1.09 1.31
#>    DEA 1.05      0.04   1 1.01   1.05 1.05 1.18

scores %>% sample_n(3)

#>     CEAT_BCC_OUT DEA_BCC_OUT
#> BLR         1.07        1.00
#> SVK         1.07        1.04
#> RUS         1.04        1.00
```

In the case of the output-oriented radial model, Esteve et al. (2021) showed how this measure can be computed through Random Forest where $\phi(\mathbf{x}_k, \mathbf{y}_k) = \max\{\phi_k \in R : (\mathbf{x}_k, \phi_k \mathbf{y}_k) \in \Psi\}$ can be estimated by substituting the theoretical production possibility set Ψ by its estimation $\hat{\Psi}_{RF+EAT}$, i.e., $\phi^{RF+EAT}(\mathbf{x}_k, \mathbf{y}_k) = \max\{\phi_k \in R : (\mathbf{x}_k, \phi_k \mathbf{y}_k) \in \hat{\Psi}_{RF+EAT}\}$. In particular, $\phi^{RF+EAT}(\mathbf{x}_k, \mathbf{y}_k)$ may be calculated as:

$$\phi^{RF+EAT}(\mathbf{x}_k, \mathbf{y}_k) = \min_{r=1,\ldots,s} \left\{ \frac{y_r^{RF+EAT(\aleph)}(\mathbf{x}_k)}{y_{rk}} \right\}, \tag{14}$$

where $y_r^{RF+EAT(\aleph)}(\mathbf{x}_k)$ is the estimation of the r-th output given the input bundle $\mathbf{x}_k$.

In R, this model can be computed using `efficiencyREAT()`:

```
scores <- efficiencyRFEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelRFEAT,
                          digits = 2, print.table = TRUE)

#>  Model Mean Std. Dev.  Min Q1 Median   Q3  Max
#>  RFEAT 1.03      0.04 0.94  1   1.02 1.02 1.15
#>    FDH 1.01      0.02 1.00  1   1.00 1.00 1.12

scores %>% sample_n(3)

#>     RFEAT_BCC_OUT FDH_BCC_OUT
#> SGP          0.94           1
#> HUN          0.99           1
#> MEX          1.00           1
```

The input-oriented radial model

By analogy with the previous section, where the output-oriented radial model was shown, it is possible to calculate the input-oriented radial technical efficiency of the input-output bundle $(\mathbf{x}_k, \mathbf{y}_k)$ by solving the following Mixed-Integer Linear Program, counterpart to (10):

$$\begin{array}{ll} \min & \theta, \\ s.t. & \\ & \sum_{i=1}^{n} \lambda_i x_{ji} \leq \theta x_{jk}, \quad j = 1, \ldots, m \\ & \sum_{i=1}^{n} \lambda_i y_{ri} \geq y_{rk}, \quad r = 1, \ldots, s \\ & \sum_{i=1}^{n} \lambda_i = 1, \\ & \lambda_i \in \{0, 1\}, \quad i = 1, \ldots, n \end{array} \tag{15}$$

The same type of technical measure can be estimated through DEA by convexification of the production frontier generated by FDH. Next, we show the Linear Programming model that should be solved in that case:

$$\begin{array}{ll} \min & \theta, \\ s.t. & \\ & \sum_{i=1}^{n} \lambda_i x_{ji} \leq \theta x_{jk}, \quad j = 1, \ldots, m \\ & \sum_{i=1}^{n} \lambda_i y_{ri} \geq y_{rk}, \quad r = 1, \ldots, s \\ & \sum_{i=1}^{n} \lambda_i = 1, \\ & \lambda_i \geq 0, \quad i = 1, \ldots, n \end{array} \tag{16}$$

The input-oriented radial model in the case of the Efficiency Analysis Trees technique can be determined through the following Mixed-Integer Linear Program:

$$\begin{array}{ll} \min & \theta, \\ s.t. & \\ & \sum_{t\in\tilde{T}^*} \lambda_t a_j^t \leq \theta x_{jk}, \qquad j = 1,...,m \\ & \sum_{t\in\tilde{T}^*} \lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk}, \quad r = 1,...,s \\ & \sum_{t\in\tilde{T}^*} \lambda_t = 1, \\ & \lambda_t \in \{0,1\}, \qquad t \in \tilde{T}^* \end{array} \tag{17}$$

In R, this model can be computed by setting `scores_model = "BCC.INP"` in `efficiencyEAT()`:

```
scores <- efficiencyEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                        scores_model = "BCC.INP", digits = 2,
                        print.table = TRUE)

#>  Model Mean Std. Dev.  Min   Q1 Median   Q3 Max
#>    EAT 0.94      0.06 0.69 0.90   0.96 0.96   1
#>    FDH 0.98      0.03 0.90 0.97   1.00 1.00   1

scores %>% sample_n(3)

#>     EAT_BCC_INP FDH_BCC_INP
#> DEU        0.88        0.92
#> KAZ        1.00        1.00
#> SRB        0.99        1.00
```

Additionally, under the Convexified Efficiency Analysis Trees technique, the optimization model corresponding to the convexification of the production possibility set derived from $conv(\hat{\Psi}_{T^*})$ from (5) should be solved in order to determine an estimation of the input-oriented radial measure as follows:

$$\begin{array}{ll} \min & \theta, \\ s.t. & \\ & \sum_{t\in\tilde{T}^*} \lambda_t a_j^t \leq \theta x_{jk}, \qquad j = 1,...,m \\ & \sum_{t\in\tilde{T}^*} \lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk}, \quad r = 1,...,s \\ & \sum_{t\in\tilde{T}^*} \lambda_t = 1, \\ & \lambda_t \geq 0, \qquad t \in \tilde{T}^* \end{array} \tag{18}$$

In R, this model can be computed by setting `scores_model = "BCC.INP"` in `efficiencyCEAT()`:

```
scores <- efficiencyCEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                         scores_model = "BCC.INP", digits = 2,
                         print.table = TRUE)

#>  Model Mean Std. Dev.  Min   Q1 Median   Q3 Max
#>   CEAT 0.82      0.08 0.69 0.76   0.81 0.81   1
#>    DEA 0.92      0.07 0.72 0.87   0.91 0.91   1

scores %>% sample_n(3)

#>     CEAT_BCC_INP DEA_BCC_INP
#> QAT         0.93        1.00
#> CYP         0.69        0.78
#> CHE         0.73        0.85
```

The output-oriented Russell measure

The output-oriented Russell measure under FDH must be calculated through the following optimization model:

$$
\begin{array}{ll}
\max & \frac{1}{s}\sum_{r=1}^{s}\phi_r, \\
s.t. & \\
& \sum_{i=1}^{n}\lambda_t x_{ji} \leq x_{jk}, \quad j=1,...,m \\
& \sum_{i=1}^{n}\lambda_t y_{ri} \geq \phi_r y_{rk}, \quad r=1,...,s \\
& \sum_{i=1}^{n}\lambda_t = 1, \\
& \lambda_i \in \{0,1\}, \quad i=1,...,n \\
& \boldsymbol{\phi} \geq \mathbf{1}_s.
\end{array}
\tag{19}
$$

Under DEA, the corresponding model would be:

$$
\begin{array}{ll}
\max & \frac{1}{s}\sum_{r=1}^{s}\phi_r, \\
s.t. & \\
& \sum_{i=1}^{n}\lambda_t x_{ji} \leq x_{jk}, \quad j=1,...,m \\
& \sum_{i=1}^{n}\lambda_t y_{ri} \geq \phi_r y_{rk}, \quad r=1,...,s \\
& \sum_{i=1}^{n}\lambda_t = 1, \\
& \lambda_i \geq 0, \quad i=1,...,n \\
& \boldsymbol{\phi} \geq \mathbf{1}_s.
\end{array}
\tag{20}
$$

If we resort to the Efficiency Analysis Trees technique, then the model to be solved should be the following:

$$
\begin{array}{ll}
\max & \frac{1}{s}\sum_{r=1}^{s}\phi_r, \\
s.t. & \\
& \sum_{t\in\tilde{T}^*}\lambda_t a_j^t \leq x_{jk}, \quad j=1,...,m \\
& \sum_{t\in\tilde{T}^*}\lambda_t d_{rT^*}(\mathbf{a}^t) \geq \phi_r y_{rk}, \quad r=1,...,s \\
& \sum_{t\in\tilde{T}^*}\lambda_t = 1, \\
& \lambda_t \in \{0,1\}, \quad i=1,...,n \\
& \boldsymbol{\phi} \geq \mathbf{1}_s.
\end{array}
\tag{21}
$$

In R, this model can be computed by setting `scores_model = "RSL.OUT"` in `efficiencyEAT()`:

```
scores <- efficiencyEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                        scores_model = "RSL.OUT", digits = 2,
                        print.table = TRUE)

scores %>% sample_n(3)
```

Finally, under the Convexified Efficiency Analysis Trees technique, the model would be:

$$
\begin{array}{ll}
\max & \frac{1}{s}\sum_{r=1}^{s}\phi_r, \\
s.t. & \\
& \sum_{t\in\tilde{T}^*}\lambda_t a_j^t \leq x_{jk}, \quad j=1,...,m \\
& \sum_{t\in\tilde{T}^*}\lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq \phi_r y_{rk}, \quad r=1,...,s \\
& \sum_{t\in\tilde{T}^*}\lambda_t = 1, \\
& \lambda_t \geq 0, \quad i=1,...,n \\
& \boldsymbol{\phi} \geq \mathbf{1}_s.
\end{array}
\tag{22}
$$

In R, this model can be computed by setting `scores_model = "RSL.OUT"` in `efficiencyCEAT()`:

```
scores <- efficiencyCEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                         scores_model = "RSL.OUT", digits = 2,
                         print.table = TRUE)

#>  Model Mean Std. Dev. Min   Q1 Median   Q3  Max
#>   CEAT 1.13      0.08   1 1.07   1.10 1.10 1.34
#>    DEA 1.06      0.05   1 1.02   1.06 1.06 1.22

scores %>% sample_n(3)

#>     CEAT_RSL_OUT DEA_RSL_OUT
#> LVA         1.07        1.05
#> CHL         1.18        1.13
#> MAR         1.14        1.00
```

The input-oriented Russell measure

By analogy with the output-oriented Russell measure, the input-oriented Russell measure should be calculated through the following optimization models, depending on the selected approach:

$$\begin{array}{ll} \min & \frac{1}{m}\sum_{j=1}^{m}\theta_j, \\ s.t. & \\ & \sum_{i=1}^{n}\lambda_t x_{ji} \leq \theta_j x_{jk}, \quad j=1,...,m \\ & \sum_{i=1}^{n}\lambda_t y_{ri} \geq y_{rk}, \quad r=1,...,s \\ & \sum_{i=1}^{n}\lambda_t = 1, \\ & \lambda_i \in \{0,1\}, \quad i=1,...n \\ & \boldsymbol{\theta} \leq \mathbf{1}_m. \end{array} \tag{23}$$

Under DEA, the corresponding model would be:

$$\begin{array}{ll} \min & \frac{1}{m}\sum_{j=1}^{m}\theta_j, \\ s.t. & \\ & \sum_{i=1}^{n}\lambda_t x_{ji} \leq \theta_j x_{jk}, \quad j=1,...,m \\ & \sum_{i=1}^{n}\lambda_t y_{ri} \geq y_{rk}, \quad r=1,...,s \\ & \sum_{i=1}^{n}\lambda_t = 1, \\ & \lambda_i \geq 0, \quad i=1,...n \\ & \boldsymbol{\theta} \leq \mathbf{1}_m. \end{array} \tag{24}$$

If we resort to the Efficiency Analysis Trees technique, then the model to be solved should be the following:

$$\begin{array}{ll} \min & \frac{1}{m}\sum_{j=1}^{m}\theta_j, \\ s.t. & \\ & \sum_{t\in\tilde{T}^*}\lambda_t a_j^t \leq \theta_j x_{jk}, \quad j=1,...,m \\ & \sum_{t\in\tilde{T}^*}\lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk}, \quad r=1,...,s \\ & \sum_{t\in\tilde{T}^*}\lambda_t = 1, \\ & \lambda_t \in \{0,1\}, \quad i=1,...,n \\ & \boldsymbol{\theta} \leq \mathbf{1}_m. \end{array} \tag{25}$$

In R, this model can be computed by setting `scores_model = "RSL.INP"` in `efficiencyEAT()`:

```
scores <- efficiencyEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                        scores_model = "RSL.INP", digits = 2,
                        print.table = TRUE)

#>  Model Mean Std. Dev.  Min   Q1 Median   Q3  Max
#>    EAT 0.58      0.09 0.43 0.52   0.56 0.56 0.81
#>    FDH 0.87      0.10 0.59 0.81   0.86 0.86 1.00

scores %>% sample_n(3)

#>     EAT_RSL_INP FDH_RSL_INP
#> LBN        0.58        0.73
#> MAR        0.69        0.97
#> MKD        0.58        0.87
```

Finally, under the Convexified Efficiency Analysis Trees technique, the model would be:

$$\begin{array}{ll} \min & \frac{1}{m}\sum_{j=1}^{m}\theta_j, \\ s.t. & \\ & \sum_{t\in\tilde{T}^*}\lambda_t a_j^t \leq \theta_j x_{jk}, \quad j=1,...,m \\ & \sum_{t\in\tilde{T}^*}\lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk}, \quad r=1,...,s \\ & \sum_{t\in\tilde{T}^*}\lambda_t = 1, \\ & \lambda_t \geq 0, \quad i=1,...,n \\ & \boldsymbol{\theta} \leq \mathbf{1}_m. \end{array} \tag{26}$$

In R, this model can be computed by setting `scores_model = "RSL.INP"` in `efficiencyCEAT()`:

```
scores <- efficiencyCEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
```

```
                        scores_model = "RSL.INP", digits = 2,
                        print.table = TRUE)

#>  Model Mean Std. Dev.  Min   Q1 Median   Q3  Max
#>   CEAT 0.54      0.08 0.43 0.49   0.53 0.53 0.79
#>    DEA 0.80      0.11 0.59 0.74   0.78 0.78 1.00

scores %>% sample_n(3)

#>     CEAT_RSL_INP DEA_RSL_INP
#> ARG         0.44        0.59
#> LUX         0.44        0.65
#> CHE         0.49        0.74
```

The directional distance function

Chambers, Chung, and Färe (1998) introduced the directional distance function (DDF) as a technical efficiency measure that projects (x_k, y_k) through a pre-assigned direction $\mathbf{g} = (-\mathbf{g}_j^-, +\mathbf{g}_r^+) \neq 0_{m+s}, \mathbf{g}_j^- \in R^m, \mathbf{g}_r^+ \in R^s$ to the efficiency frontier of the corresponding technology. Under FDH, the DDF is calculated as follows:

$$\begin{array}{lll} \max & \beta_k, & \\ s.t. & & \\ & \sum_{i=1}^n \lambda_i x_{ji} \leq x_{jk} - \beta_k g_j^-, & j = 1, ..., m \\ & \sum_{i=1}^n \lambda_i y_{ri} \geq y_{rk} + \beta_k g_r^+, & r = 1, ..., s \\ & \sum_{i=1}^n \lambda_i = 1, & \\ & \lambda_i \in \{0, 1\}, & i = 1, ..., n \end{array} \tag{27}$$

The corresponding linear program in DEA is as follows:

$$\begin{array}{lll} \max & \beta_k, & \\ s.t. & & \\ & \sum_{i=1}^n \lambda_i x_{ji} \leq x_{jk} - \beta_k g_j^-, & j = 1, ..., m \\ & \sum_{i=1}^n \lambda_i y_{ri} \geq y_{rk} + \beta_k g_r^+, & r = 1, ..., s \\ & \sum_{i=1}^n \lambda_i = 1, & \\ & \lambda_i \geq 0, & i = 1, ..., n \end{array} \tag{28}$$

In the context of Efficiency Analysis Trees, the DDF is calculated through the following Mixed-Integer Linear Program:

$$\begin{array}{lll} \max & \beta_k, & \\ s.t. & & \\ & \sum_{t \in \tilde{T}^*} \lambda_t a_j^t \leq x_{jk} - \beta_k g_j^-, & j = 1, ..., m \\ & \sum_{t \in \tilde{T}^*} \lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk} + \beta_k g_r^+, & r = 1, ..., s \\ & \sum_{t \in \tilde{T}^*} \lambda_t = 1, & \\ & \lambda_t \in \{0, 1\}, & t \in \tilde{T}^*. \end{array} \tag{29}$$

In R, this model can be computed by setting `scores_model = "DDF"` in `efficiencyEAT()`:

```
scores <- efficiencyEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                        scores_model = "DDF", digits = 2,
                        print.table = TRUE)

#>  Model Mean Std. Dev. Min Q1 Median   Q3  Max
#>    EAT 0.02      0.02   0  0   0.01 0.01 0.13
#>    FDH 0.01      0.01   0  0   0.00 0.00 0.05

scores %>% sample_n(3)

#>     EAT_DDF FDH_DDF
#> MDA    0.00    0.00
#> LVA    0.00    0.00
#> BRA    0.03    0.01
```

In the case of Convexified Efficiency Analysis Trees, the optimization model is as follows.

$$\begin{array}{ll} \max & \beta_k, \\ s.t. & \\ & \sum_{t\in\tilde{T}^*}\lambda_t a_j^t \leq x_{jk} - \beta_k g_j^-, \qquad j = 1,...,m \\ & \sum_{t\in\tilde{T}^*}\lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk} + \beta_k g_r^+, \quad r = 1,...,s \\ & \sum_{t\in\tilde{T}^*}\lambda_t = 1, \\ & \lambda_t \geq 0, \qquad t \in \tilde{T}^*. \end{array} \tag{30}$$

In R, this model can be computed by setting `scores_model = "DDF"` in `efficiencyCEAT()`:

```
scores <- efficiencyCEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                         scores_model = "DDF", digits = 2,
                         print.table = TRUE)

#>  Model Mean Std. Dev. Min   Q1 Median   Q3  Max
#>   CEAT 0.07      0.04   0 0.05   0.06 0.06 0.18
#>    DEA 0.03      0.03   0 0.00   0.03 0.03 0.12

scores %>% sample_n(3)

#>     CEAT_DDF DEA_DDF
#> IRL     0.05    0.03
#> LBN     0.15    0.10
#> FRA     0.07    0.06
```

The weighted additive model

The additive model measures technical efficiency based on input excesses and output shortfalls. It characterizes efficiency in terms of the input and output slacks: $\mathbf{s}^- \in R^m$ and $\mathbf{s}^+ \in R^s$, respectively. The **eat** package implements the weighted additive model formulation of Lovell and Pastor (1995), where $(\mathbf{w}^-, \mathbf{w}^+) \in R_+^m \times R_+^s$ are the input and output weights whose elements can vary across DMUs.

In the case of the FDH, the optimization program to be solved would be:

$$\begin{array}{ll} \max & \sum_{j=1}^m w_j^- s_{jk}^- + \sum_{r=1}^s w_r^+ s_{rk}^+, \\ s.t. & \\ & \sum_{i=1}^n \lambda_t x_{ji} \leq x_{jk} - s_{jk}^-, \qquad j = 1,...,m \\ & \sum_{i=1}^n \lambda_t y_{ri} \geq y_{rk} + s_{rk}^+, \qquad r = 1,...,s \\ & \sum_{i=1}^n \lambda_i = 1, \\ & \lambda_i \in \{0,1\}, \qquad i = 1,...,n \\ & \mathbf{s}_k^- \geq \mathbf{0}_m, \mathbf{s}_k^+ \geq \mathbf{0}_s. \end{array} \tag{31}$$

Under DEA, the model would be as follows:

$$\begin{array}{ll} \max & \sum_{j=1}^m w_j^- s_{jk}^- + \sum_{r=1}^s w_r^+ s_{rk}^+, \\ s.t. & \\ & \sum_{i=1}^n \lambda_t x_{ji} \leq x_{jk} - s_{jk}^-, \qquad j = 1,...,m \\ & \sum_{i=1}^n \lambda_t y_{ri} \geq y_{rk} + s_{rk}^+, \qquad r = 1,...,s \\ & \sum_{i=1}^n \lambda_i = 1, \\ & \lambda_i \geq 0, \qquad i = 1,...,n \\ & \mathbf{s}_k^- \geq \mathbf{0}_m, \mathbf{s}_k^+ \geq \mathbf{0}_s. \end{array} \tag{32}$$

Within the framework of Efficiency Analysis Trees, the weighted additive model would be calculated as follows:

$$\begin{array}{ll} \max & \sum_{j=1}^m w_j^- s_{jk}^- + \sum_{r=1}^s w_r^+ s_{rk}^+, \\ s.t. & \\ & \sum_{t\in\tilde{T}^*}\lambda_t a_j^t \leq x_{jk} - s_{jk}^-, \qquad j = 1,...,m \\ & \sum_{t\in\tilde{T}^*}\lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk} + s_{rk}^+, \quad r = 1,...,s \\ & \sum_{i=1}^n \lambda_i = 1, \\ & \lambda_i \in \{0,1\}, \qquad i = 1,...,n \\ & \mathbf{s}_k^- \geq \mathbf{0}_m, \mathbf{s}_k^+ \geq \mathbf{0}_s. \end{array} \tag{33}$$

In R, this model can be computed by setting `scores_model = "WAM.MIP"` for the Measure of Inefficiency Proportions or `"WAM.RAM"` for the Range-Adjusted Measure of Inefficiency (W. W. Cooper, Park, and Pastor 1999) in `efficiencyEAT()`:

```
scores <- efficiencyEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                        scores_model = "WAM.MIP", digits = 2,
                        print.table = TRUE)

#>  Model Mean Std. Dev.  Min   Q1 Median   Q3  Max
#>    EAT 2.07      0.56 0.61 1.58   2.18 2.18 3.14
#>    FDH 0.40      0.58 0.00 0.00   0.00 0.00 2.37

scores %>% sample_n(3)

#>     EAT_WAM_MIP FDH_WAM_MIP
#> MLT        2.54           0
#> SVN        1.86           0
#> HRV        2.30           0

scores <- efficiencyEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                        scores_model = "WAM.RAM", digits = 2,
                        print.table = TRUE)

#>  Model    Mean Std. Dev.   Min      Q1  Median      Q3     Max
#>    EAT 2704.65   1677.11 455.3 1467.78 2425.09 2425.09 8293.59
#>    FDH  959.09   1388.56   0.0    0.00    0.00    0.00 5413.27

scores %>% sample_n(3)

#>     EAT_WAM_RAM FDH_WAM_RAM
#> ITA     2192.98        0.00
#> LVA     1890.38        0.00
#> CAN     1724.26     1182.66
```

And, finally, the Convexified Efficiency Analysis Trees weighted additive model would be:

$$
\begin{aligned}
\max \quad & \textstyle\sum_{j=1}^{m} w_j^- s_{jk}^- + \sum_{r=1}^{s} w_r^+ s_{rk}^+, \\
s.t. \quad & \\
& \textstyle\sum_{t \in \tilde{T}^*} \lambda_t a_j^t \leq x_{jk} - s_{jk}^-, && j = 1, ..., m \\
& \textstyle\sum_{t \in \tilde{T}^*} \lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk} + s_{rk}^+, && r = 1, ..., s \\
& \textstyle\sum_{i=1}^{n} \lambda_i = 1, & & \\
& \lambda_i \geq 0, && i = 1, ..., n \\
& \mathbf{s}_k^- \geq \mathbf{0}_m, \mathbf{s}_k^+ \geq \mathbf{0}_s. &&
\end{aligned}
\tag{34}
$$

In R, this model can be computed by setting `scores_model = "WAM.MIP"` for the Measure of Inefficiency Proportions or `"WAM.RAM"` for the Range-Adjusted Measure of Inefficiency in `efficiencyCEAT()`:

```
scores <- efficiencyCEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                         scores_model = "WAM.MIP", digits = 2,
                         print.table = TRUE)

#>  Model Mean Std. Dev.  Min   Q1 Median   Q3  Max
#>   CEAT 2.39      0.45 0.96 2.24   2.50 2.50 3.14
#>    DEA 0.90      0.62 0.00 0.23   1.07 1.07 2.37

scores %>% sample_n(3)

#>     CEAT_WAM_MIP DEA_WAM_MIP
#> ISR         2.74        1.13
#> ITA         2.73        1.31
#> SGP         1.91        0.00
```

```
scores <- efficiencyCEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                         scores_model = "WAM.RAM", digits = 2,
                         print.table = TRUE)

#>  Model    Mean Std. Dev.    Min      Q1  Median      Q3      Max
#>   CEAT 5285.99   2573.13 612.54 3362.98 4836.79 4836.79 12088.64
#>    DEA 2413.66   1821.33   0.00  746.22 2495.20 2495.20  7167.29

scores %>% sample_n(3)

#>     CEAT_WAM_RAM DEA_WAM_RAM
#> KOR      1538.61        0.00
#> ROU      7239.16     4727.17
#> EST       612.54        0.00
```

6 Advanced options and displaying and exporting results

Advanced optimization options

The `bestEAT()` and `bestRFEAT()` functions are aimed at finding the value of the hyperparameters that minimize the root mean squared error (RMSE) calculated from a test sample through an Efficiency Analysis Trees or a Random Forest for Efficiency Analysis Trees model fitted using a training sample. The code of these functions is as follows:

```
# Hyperparameter tuning for Efficiency Analysis Trees
bestEAT(
  training, test, x, y,
  numStop = 5, fold = 5,
  max.depth = NULL,
  max.leaves = NULL,
  na.rm = TRUE
  )

# Hyperparameter tuning for Random Forest for Efficiency Analysis Trees
bestRFEAT(
  training, test, x, y,
  numStop = 5, m = 50,
  s_mtry = c("5", "BRM"),
  na.rm = TRUE
  )
```

Here is an example of using the `bestEAT()` function. First, the PISAindex database explained in Section Data structure is divided into a training subset with 70% of the DMUs and a test subset with the remaining 30% (these values can be modified).

```
n <- nrow(PISAindex)              # Observations in the dataset
selected <- sample(1:n, n * 0.7)  # Training indexes
training <- PISAindex[selected, ] # Training set
test <- PISAindex[- selected, ]   # Test set
```

Then, we can apply the `bestEAT()` function. This function, and its equivalent `bestRFEAT()`, requires a training set (`training`) on which to fit an Efficiency Analysis Trees model (with cross-validation), a test set (`test`) on which to calculate the root mean squared error and the input and output indexes (`x` and `y`, respectively). The rest of the arguments (`numStop`, `fold`, `max.depth` and `max.leaves` in case of using the `bestEAT()` function) are used to create a grid of combinations that determines the number of models to fit. Notice that it is not possible to enter `NULL` and a certain value in `max.depth` or `max.leaves` arguments at the same time (i.e. `max.depth = c(NULL, 5, 3)`).

In the following example, the arguments `numStop = (3, 5, 7)` and `fold = (5, 7)` are entered and, consequently, six different models are constructed and fitted with {`numStop = 3`, `fold = 5`}, {`numStop = 3`, `fold = 7`}, {`numStop = 5`, `fold = 5`}, {`numStop = 5`, `fold = 7`}, {`numStop = 7`, `fold = 5`} and {`numStop = 7`, `fold = 7`}. Let us show a numerical example:

```
bestEAT(training = training, test = test, x = 4:8, y = 1:3,
        numStop = c(3, 5, 7), fold = c(5, 7))

#>   numStop fold   RMSE leaves
#> 1       5    5  74.74     15
#> 2       7    7  83.61     13
#> 3       3    5  84.17     13
#> 4       3    7  84.17     13
#> 5       7    5  86.39      8
#> 6       5    7 104.93      8
```

The best model is given by the hyperparameters {`numStop = 5`, `fold = 5`} with RMSE = 74.74 and 15 leaf nodes. Note that sometimes it might be interesting to select a model with a higher RMSE but with a lower number of leaf nodes. With this result, we fit the final Efficiency Analysis Trees model using all the original data.

```
bestEAT_model <- EAT(data = PISAindex, x = 4:8, y = 1:3, numStop = 5, fold = 5)
```

Displaying results

General functions for the EAT object

The simplest functions to use in order to explore the results of an `EAT` object are `print()` and `summary()`. The function `print()` returns the tree-structure of an Efficiency Analysis Trees model; while the function `summary()` returns general information about the fitted model. We show the results with an example:

```
modelEAT2 <- EAT(data = PISAindex, x = 7, y = 3)

print(modelEAT2) # [node] y: [prediction] || R: error n(t): nº of DMUs

#>  [1] y: [ 569 ] || R: 15724.19 n(t): 72
#>
#>  |  [2] AAE < 70.12 --> y: [ 486 ] || R: 3094.43 n(t): 34
#>
#>  |   |  [4] AAE < 60.75 --> y: [ 472 ] <*> || R: 1553.93 n(t): 17
#>
#>  |   |  [5] AAE >= 60.75 --> y: [ 486 ] <*> || R: 1006.94 n(t): 17
#>
#>  |  [3] AAE >= 70.12 --> y: [ 569 ] <*> || R: 3753.38 n(t): 38
#>
#> <*> is a leaf node

# Primary & surrogate splits: Node i --> {SL, SR} || var --> {R: error, s: threshold}
summary(modelEAT2)

#>
#>   Formula:  M_PISA ~ AAE
#>
#>  # ========================== #
#>  #   Summary for leaf nodes   #
#>  # ========================== #
#>
#>  id n(t)  % M_PISA    R(t)
#>   3   38 53    569 3753.38
#>   4   17 24    472 1553.93
#>   5   17 24    486 1006.94
#>
#>  # ========================== #
#>  #            Tree            #
#>  # ========================== #
#>
#>  Interior nodes: 2
```

```
#>      Leaf nodes: 3
#>     Total nodes: 5
#>
#>            R(T): 6314.25
#>         numStop: 5
#>            fold: 5
#>       max.depth:
#>      max.leaves:
#>
#>  # ========================== #
#>  # Primary & surrogate splits #
#>  # ========================== #
#>
#>  Node 1 --> {2,3} || AAE --> {R: 6847.81, s: 70.12}
#>
#>  Node 2 --> {4,5} || AAE --> {R: 2560.88, s: 60.75}
```

Representing the efficiency scores

`efficiencyJitter()` returns a jitter plot from `ggplot2`. This graphic shows how DMUs are grouped into leaf nodes in a model built using the `EAT()` function where each leaf node groups DMUs with the same level of resources. A black dot and a black line represent, respectively, the mean value and the standard deviation of the scores (`df_scores` from the `efficiencyEAT()` or the `efficiencyCEAT()` functions) of a given node. Additionally, efficient DMU labels are always displayed based on the model entered in the `scores_model` argument. Finally, the user can specify an upper bound (`upb`) and a lower bound (`lwb`) in order to show, in addition, the labels whose efficiency score lies between them. The code is as follows:

```
efficiencyJitter(
  object,
  df_scores,
  scores_model,
  upb = NULL,
  lwb = NULL
)
```

As an example, using data from Section Data structure, we create a new Efficiency Analysis Trees model containing only the `AAE` and the `M_PISA` variables. Next, we evaluate the Efficiency Analysis Trees efficiency scores corresponding to the output-oriented radial model and plot them through `efficiencyJitter()`.

```
scores <- efficiencyEAT(data = PISAindex, x = 7, y = 3, object = modelEAT2,
                        scores_model = "BCC.OUT", digits = 2,
                        print.table = FALSE)
```

`efficiencyDensity()` returns a density plot from `ggplot2`. This graphic allows to verify the similarity between the scores obtained by the different available methodologies (`EAT`, `FDH`, `CEAT`, `DEA` and `RFEAT`) in the **eat** package.

```
efficiencyDensity(
  df_scores,
  model = c("EAT", "FDH")
)
```

In this case, a comparison between the scores of the EAT and FDH models is shown, where it can be clearly seen how FDH is less restrictive when determining a unit as efficient:

Other graphics

In the limited case of using only one input for producing only one output, we can display the frontier (from `ggplot2`) estimated by the `EAT()` function through the `frontier()` function:

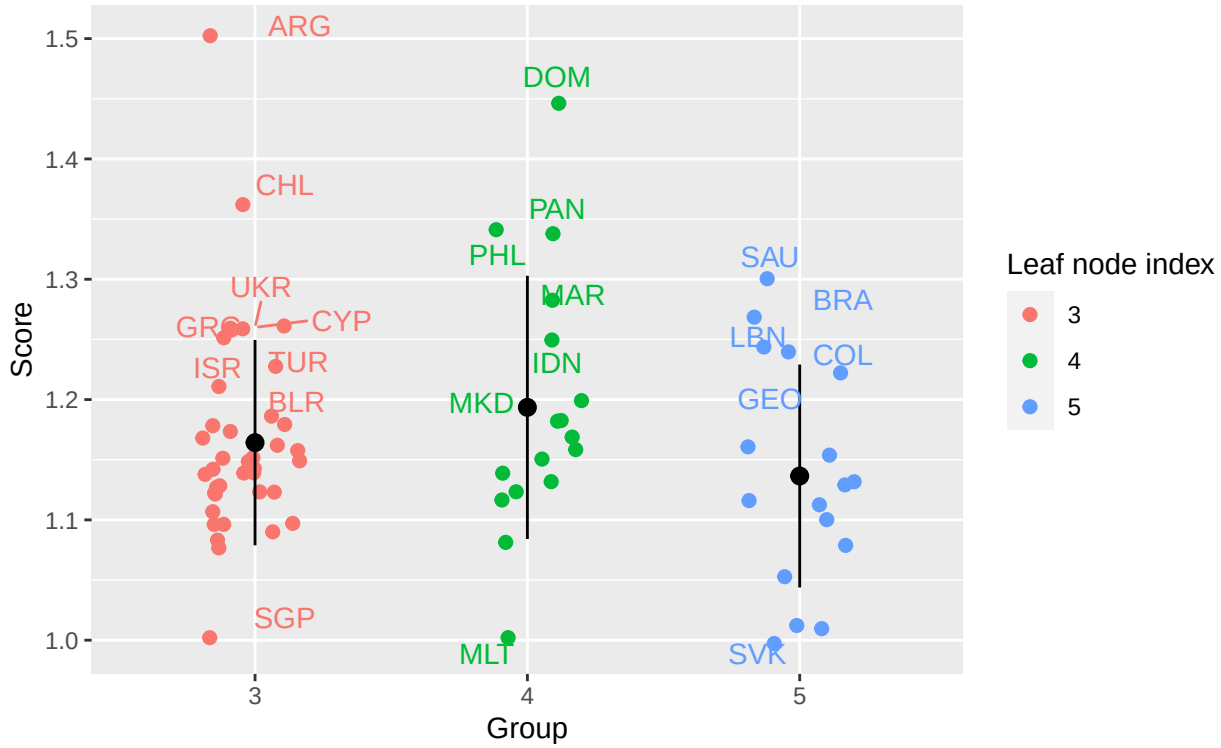

Figure 3: Jitter plot generated by 'efficiencyJitter()' to show how the countries are grouped inside three particular leaf nodes.

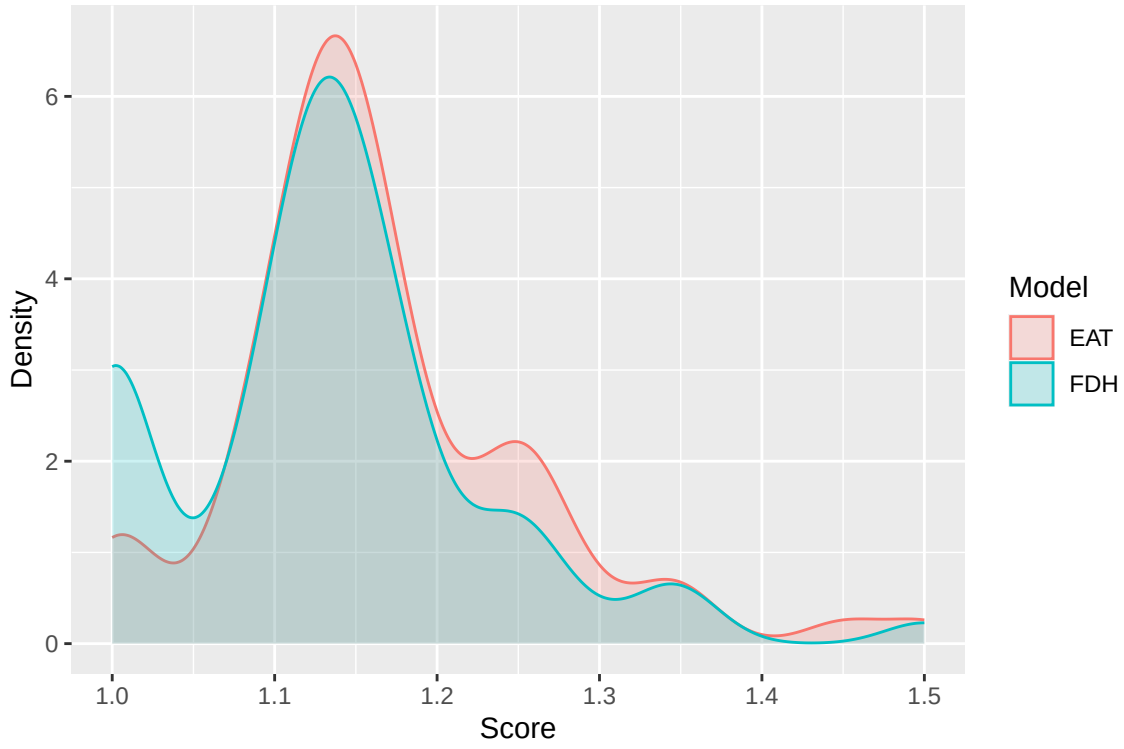

Figure 4: Density plot generated by 'efficiencyDensity()' to show the difference between the score obtained by EAT and FDH.

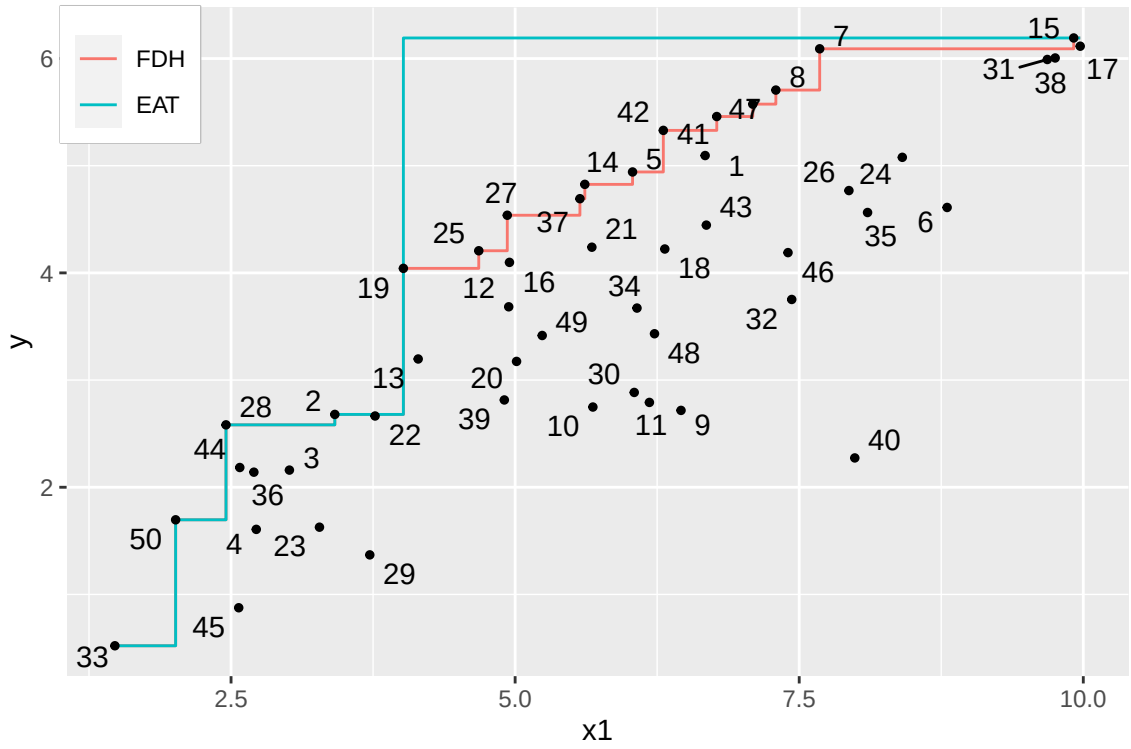

Figure 5: Plot of productions functions corresponding to the EAT and the FDH estimator when 'frontier()' is applied.

```
frontier(
  object, FDH = TRUE,
  observed.data = TRUE,
  observed.color = "black",
  pch = 19,vsize = 1,
  rwn = FALSE,
  max.overlaps = 10
)
```

Optionally, the frontier estimated by FDH can also be plotted if `FDH = TRUE`. Observed DMUs can be showed by a scatterplot if `observed.data = TRUE` and its color, shape and size can be modified with `observed.color`, `pch` and `size` respectively. Finally, row names can be included with `rwn = TRUE`.

As an example, we use data simulated from the **eat** package to generate a `data.frame` with 50 rows (`N` = DMUs) and 1 input (`nX`):

```
simulated <- Y1.sim(N = 50, nX = 1)
modelEAT3 <- EAT(data = simulated, x = 1, y = 2)
```

Then, we apply the `frontier()` function, where it can be observed how the Efficiency Analysis Trees model generalizes the results obtained by the FDH model:

The function `frontier()` shown above only works for the simple case of a low-dimensional scenario with one input and one output. For multiple input and/or output scenarios, the typical tree-structure showing the relationships between outputs and inputs is given by the function `plotEAT()`.

```
plotEAT(
  object
)
```

The nodes of the tree are colored according to the variable by which the split is performed or they are black, in the case of being a leaf node. For each node, we can obtain the following information:

- `id`: node index.

- `R`: error at the node.
- `n(t)`: number of DMUs at the node.
- input variable associated with the split.
- `y`: vector of output predictions.

Next, we limit the growth of an Efficiency Analysis Trees model to a maximum size of 5 (`max.depth = 4`) and display the tree-structure using the `plotEAT()` function:

Finally, the function `plotRFEAT()` returns the Out-Of-Bag error for a random forest consisting of k trees. The code of the function and an example with the object `modelRFEAT` are shown above:

```
plotRFEAT(
  object
)
```

In view of the results, it can be seen how the OOB error presents a great variability for a small number of trees, however, it usually levels off. In our case, it seems that the OOB error levels off from 20 trees onwards around an OOB error of 56, so it could be interesting not to include a greater number of trees in the random forest in order to reduce the computational cost.

7 Conclusions

The **eat** package allows the estimation of production frontiers in microeconomics and engineering through suitable adaptations of Regression Trees and Random Forest. In the first case, the package implements in R the so-called Efficiency Analysis Trees (EAT) by Esteve et al. (2020), which is a non-parametric technique that competes against the more standard Free Disposal Hull (FDH) technique. In this regard, the EAT technique overcomes the overfitting problem suffered by the FDH technique. FDH is based on three microeconomic postulates. First, the technology determined by FDH satisfies free disposability in inputs and outputs. Second, it is assumed to be deterministic, that is, the production possibility set built by this technique always contains all the observations that belong to the data sample. Third, FDH meets the minimal extrapolation principle. This last postulate implies that FDH generates the smallest set that satisfies the first two postulates. Consequently, the derived efficient frontier is as close to the data as possible, generating overfitting problems. In contrast, the Efficiency Analysis Trees (EAT) technique meets the first two postulates but does not satisfy the minimal extrapolation principle. This fact avoids possible overfitting problems. The difficulty for non-overfitted models lies in where to locate the production possibility set in such a way that it is close to the (unknown) technology associated with the underlying Data Generating Process. In the case of EAT, it is achieved through cross-validation and pruning. A subsequent convexification of the EAT estimation of the technology, known as CEAT by its acronym, yields an alternative estimate of the production possibility set in contrast to the traditional Data Envelopment Analysis (DEA) technique. In the second case, an ensemble of tree models is fitted and aggregated with the objective of achieving robustness in the estimation of the production frontier (Esteve et al. 2021).

Several functions have been implemented in the **eat** package for determining the best model, through a pruning process based on cross-validation, graphing the results, calculating a ranking of importance of inputs and comparing the efficiency scores estimated by EAT with respect to the standard approaches, i.e., FDH and DEA, through a list of standard technical efficiency measures. We refer to the input and output-oriented radial models, the input and output-oriented Russell measures, the Directional Distance Function and the Weighted Additive model.

Throughout the paper, we have also shown how to organize the data, use the available functions, and interpret the results. In particular, to illustrate the different functions implemented in the package, we applied all of them on a common empirical example so that results can easily be compared. In this way, we believe that the **eat** package is a valid self-contained R package for the measurement of technical efficiency from the popular machine learning technique: Decision Trees. Finally, since the code is freely available in an open source repository, users will benefit from the collaboration and review of the community. Users may check and modify the code to adapt it to their own needs and extend it with new definitions.

8 Acknowledgments

M. Esteve, V. España and J. Aparicio thank the grant PID2019-105952GB-I00 funded by Ministerio de Ciencia e Innovación/ Agencia Estatal de Investigación /10.13039/501100011033. Additionally, M.

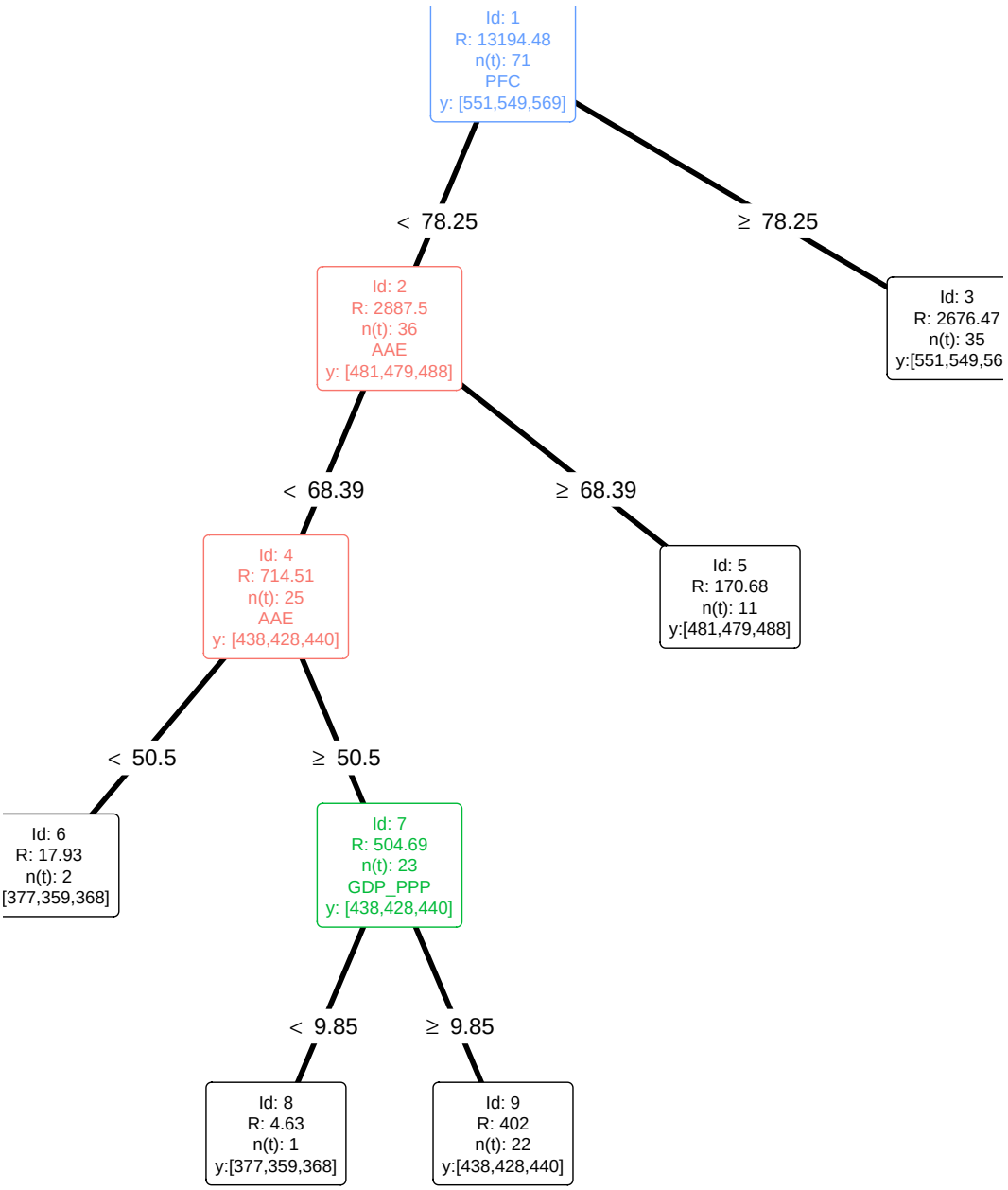

Figure 6: Plot of the tree structure obtained through an EAT model with the parameter max.depth defined as 4.

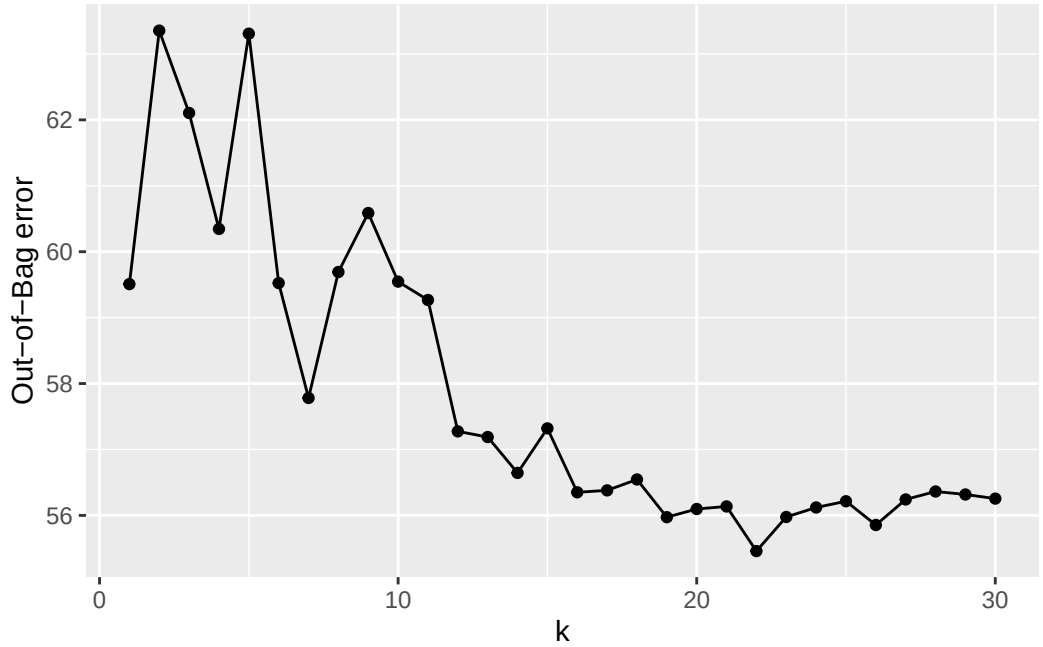

Figure 7: Plot of the OOB error corresponding to 30 different RFEAT where k represents the number of trees belonging to each RF.

Esteve gratefully acknowledges the financial support from the Spanish Ministry of Science, Innovation and Universities under Grant FPU17/05365. X. Barber gratefully acknowledges the financial support from the Spanish Ministry of Science and the State Research Agency under grant PID2019-106341GB-I00. X Barber and J. Aparicio gratefully acknowledge the financial support from the University Miguel Hernandez and the Vice-Rectorate for Research under grant AW1020IP-2020/NAC/00073. This work was also supported by the Generalitat Valenciana under Grant ACIF/2021 (V. España).

References

Afriat, Sidney N. 1972. "Efficiency Estimation of Production Functions." *International Economic Review*, 568–98.

Álvarez, Inmaculada, Javier Barbero, and José Zofio. 2020. "A Data Envelopment Analysis Toolbox for MATLAB." *Journal of Statistical Software (Online)* 95 (3).

Aparicio, Juan, Miriam Esteve, Jesus J Rodriguez-Sala, and José Zofio. 2021. "The Estimation of Productive Efficiency Through Machine Learning Techniques: Efficiency Analysis Trees." In *Data-Enabled Analytics: DEA for Big Data*, edited by Joe Zhu and Vicent Charles. Springer.

Aparicio, Juan, Jesús T Pastor, Fernando Vidal, and José L Zof 'io. 2017. "Evaluating Productive Performance: A New Approach Based on the Product-Mix Problem Consistent with Data Envelopment Analysis." *Omega* 67: 134–44.

Arnaboldi, Michela, Giovanni Azzone, and Marco Giorgino. 2014. *Performance Measurement and Management for Engineers*. Academic Press.

Balk, Bert M, Javier Barbero, and José L Zof 'Io. 2018. "A Total Factor Productivity Toolbox for MATLAB." *Available at SSRN 3178911*.

Banker, Rajiv D. 1993. "Maximum Likelihood, Consistency and Data Envelopment Analysis: A Statistical Foundation." *Management Science* 39 (10): 1265–73.

———. 1996. "Hypothesis Tests Using Data Envelopment Analysis." *Journal of Productivity Analysis* 7 (2): 139–59.

Banker, Rajiv D, Abraham Charnes, and William Wager Cooper. 1984. "Some Models for Estimating Technical and Scale Inefficiencies in Data Envelopment Analysis." *Management Science* 30 (9): 1078–92.

Banker, Rajiv D, Abraham Charnes, William W Cooper, and Roger Clarke. 1989. "Constrained Game Formulations and Interpretations for Data Envelopment Analysis." *European Journal of Operational Research* 40 (3): 299–308.

Banker, Rajiv D, and Ajay Maindiratta. 1992. "Maximum Likelihood Estimation of Monotone and Concave Production Frontiers." *Journal of Productivity Analysis* 3 (4): 401–15.

Berk, Richard A. 2016. *Statistical Learning from a Regression Perspective*. Vol. 14. Springer.
Bogetoft, Peter, and Lars Otto. 2010. *Benchmarking with DEA, SFA, and r*. Vol. 157. Springer Science & Business Media.
Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32.
Breiman, Leo, Jerome Friedman, Charles J Stone, and Richard A Olshen. 1984. *Classification and Regression Trees*. CRC press.
Chambers, Robert G, Yangho Chung, and Rolf Färe. 1998. "Profit, Directional Distance Functions, and Nerlovian Efficiency." *Journal of Optimization Theory and Applications* 98 (2): 351–64.
Charles, Vincent, Juan Aparicio, and Joe Zhu. 2020. *Data Science and Productivity Analytics*. Springer.
Charnes, Abraham, William W Cooper, and Edwardo Rhodes. 1978. "Measuring the Efficiency of Decision Making Units." *European Journal of Operational Research* 2 (6): 429–44.
Cooper, William W, Kyung Sam Park, and Jesus T Pastor. 1999. "RAM: A Range Adjusted Measure of Inefficiency for Use with Additive Models, and Relations to Other Models and Measures in DEA." *Journal of Productivity Analysis* 11 (1): 5–42.
Cooper, WW, LM Seiford, K Tone, and J Zhu. 2007. "Some Models and Measures for Evaluating Performances with DEA: Past Accomplishments and Future Prospects." *Journal of Productivity Analysis* 28 (3): 151–63.
Daouia, Abdelaati, Thibault Laurent, and Hohsuk Noh. 2017. "npbr: A Package for Nonparametric Boundary Regression in R." *Journal of Statistical Software* 79 (9): 1–43. https://doi.org/10.18637/jss.v079.i09.
Daraio, Cinzia, and Léopold Simar. 2005. "Introducing Environmental Variables in Nonparametric Frontier Models: A Probabilistic Approach." *Journal of Productivity Analysis* 24 (1): 93–121.
Debreu, Gerard. 1951. "The Coefficient of Resource Utilization." *Econometrica: Journal of the Econometric Society*, 273–92.
Deprins, D, and L Simar. 1984. "Measuring Labor Efficiency in Post Offices, the Performance of Public Enterprises: Concepts and Measurements, M. Marchand, P. Pestieau and H. Tulkens." Amsterdam, North (Holland, 243 (267.
Dyson, Robert G, Rachel Allen, Ana S Camanho, Victor V Podinovski, Claudia S Sarrico, and Estelle A Shale. 2001. "Pitfalls and Protocols in DEA." *European Journal of Operational Research* 132 (2): 245–59.
Efron, Brad. 1979. "Bootstrap Methods: Another Look at the Jackknife." *Annals of Statistics* 7 (1): 1–26.
Esteve, Miriam, Juan Aparicio, Alejandro Rabasa, and Jesus J Rodriguez-Sala. 2020. "Efficiency Analysis Trees: A New Methodology for Estimating Production Frontiers Through Decision Trees." *Expert Systems with Applications* 162: 113783.
Esteve, Miriam, Juan Aparicio, Jesus J Rodriguez-Sala, and Joe Zhu. 2021. "Estimation of Production Frontiers Through Random Forest: The Treatment of Lack of Robustness, Ranking of Inputs and Curse of Dimensionality Under Free Disposal Hull. Working Paper." Working paper.
Färe, Rolf, and CA Knox Lovell. 1978. "Measuring the Technical Efficiency of Production." *Journal of Economic Theory* 19 (1): 150–62.
Färe, Rolf, and Daniel Primont. 1995. *Multi-Output Production and Duality: Theory and Applications*. Springer Science & Business Media.
Farrel, MJ. 1957. "The Measure of Productive Efficiency." *Journal of the Royal Statistical Society* 120.
Ferrara, Giancarlo, and Francesco Vidoli. 2018. *Semsfa: Semiparametric Estimation of Stochastic Frontier Models*. https://CRAN.R-project.org/package=semsfa.
Friedman, Leo, and Zilla Sinuany Stern. 1998. "Combining Ranking Scales and Selecting Variables in the DEA Context: The Case of Industrial Branches." *Computers & Operations Research* 25 (9): 781–91.
Golany, Boaz, and Yaakov Roll. 1989. "An Application Procedure for DEA." *Omega* 17 (3): 237–50.
Homburg, Carsten. 2001. "Using Data Envelopment Analysis to Benchmark Activities." *International Journal of Production Economics* 73 (1): 51–58.
Ji, Yong-bae, and Choonjoo Lee. 2010. "Data Envelopment Analysis." *The Stata Journal* 10 (2): 267–80.
Khezrimotlagh, Dariush, Joe Zhu, Wade D Cook, and Mehdi Toloo. 2019. "Data Envelopment Analysis and Big Data." *European Journal of Operational Research* 274 (3): 1047–54.
Koopmans, Tjalling C. 1951. "An Analysis of Production as an Efficient Combination of Activities." *Activity Analysis of Production and Allocation*.
Kuhn, Max, Kjell Johnson, et al. 2013. *Applied Predictive Modeling*. Vol. 26. Springer.
Kuosmanen, Timo, and Andrew Johnson. 2017. "Modeling Joint Production of Multiple Outputs in StoNED: Directional Distance Function Approach." *European Journal of Operational Research* 262 (2): 792–801.
Kuosmanen, Timo, and Andrew L Johnson. 2010. "Data Envelopment Analysis as Nonparametric Least-Squares Regression." *Operations Research* 58 (1): 149–60.
Landete, Mercedes, Juan F Monge, and José L Ruiz. 2017. "Robust DEA Efficiency Scores: A Probabilistic/Combinatorial Approach." *Expert Systems with Applications* 86: 145–54.
Lovell, CA Knox, and Jesús T Pastor. 1995. "Units Invariant and Translation Invariant DEA Models."

Operations Research Letters 18 (3): 147–51.

McKenzie, Taylor. 2018. *Snfa: Smooth Non-Parametric Frontier Analysis*. https://CRAN.R-project.org/package=snfa.

Nunamaker, Thomas R. 1985. "Using Data Envelopment Analysis to Measure the Efficiency of Non-Profit Organizations: A Critical Evaluation." *Managerial and Decision Economics* 6 (1): 50–58.

O'Donnell, Christopher J et al. 2018. *Productivity and Efficiency Analysis*. Springer.

OECD, PISA. 2018. "Assessment and Analytical Framework, PISA." OECD Publishing. Paris., PISA (OECD, 2019).

Oh, Dong-hyun, and Dukrok Suh. 2013. *Nonparaeff: Nonparametric Methods for Measuring Efficiency and Productivity*. https://CRAN.R-project.org/package=nonparaeff.

Orea, Luis, and José Luis Zof 'io. 2019. "Common Methodological Choices in Nonparametric and Parametric Analyses of Firms' Performance." In *The Palgrave Handbook of Economic Performance Analysis*, 419–84. Springer.

Pastor, Jesús T, JosÉ L Ruiz, and Inmaculada Sirvent. 2002. "A Statistical Test for Nested Radial DEA Models." *Operations Research* 50 (4): 728–35.

Raab, Raymond L, and Richard W Lichty. 2002. "Identifying Subareas That Comprise a Greater Metropolitan Area: The Criterion of County Relative Efficiency." *Journal of Regional Science* 42 (3): 579–94.

Shephard, Ronald William. 1953. *Theory of Cost and Production Functions*. Princeton University Press.

Simar, Leopold, and Paul W Wilson. 1998. "Sensitivity Analysis of Efficiency Scores: How to Bootstrap in Nonparametric Frontier Models." *Management Science* 44 (1): 49–61.

———. 2000. "Statistical Inference in Nonparametric Frontier Models: The State of the Art." *Journal of Productivity Analysis* 13 (1): 49–78.

Simar, Léopold, and Paul W Wilson. 2000. "A General Methodology for Bootstrapping in Non-Parametric Frontier Models." *Journal of Applied Statistics* 27 (6): 779–802.

Social Progress Index. 2018. "Social Progress Index 2018." https://www.socialprogress.org/.

StataCorp. 2021. "Stata Statistical Software: Release 14." http://www.stata.com/.

The MathWorks Inc. 2021. "MATLAB – the Language of Technical Computing." http://www.mathworks.com/products/matlab/.

Zhu, Joe et al. 2019. "DEA Under Big Data: Data Enabled Analytics and Network Data Envelopment Analysis." *Annals of Operations Research*, 1–23.

Miriam Esteve
Miguel Hernandez University
Center of Operations Research
03202 Elche, Spain
https://cio.umh.es/
ORCiD: 0000-0002-5908-0581
miriam.estevec@umh.es

Victor España
Miguel Hernandez University
Center of Operations Research
03202 Elche, Spain
https://cio.umh.es/
ORCiD: 0000-0002-1807-6180
vespana@umh.es

Juan Aparicio
Miguel Hernandez University
Center of Operations Research
03202 Elche, Spain
https://cio.umh.es/
ORCiD: 0000-0002-0867-0004
j.aparicio@umh.es

Xavier Barber
Miguel Hernandez University
Center of Operations Research
03202 Elche, Spain
https://cio.umh.es/
ORCiD: 0000-0003-3079-5855
xbarber@umh.es