

# Package ‘Benchmarking’

May 10, 2018

**Type** Package

**Title** Benchmark and Frontier Analysis Using DEA and SFA

**Version** 0.27

**Date** 2018-5-10

**Author** Peter Bogetoft and Lars Otto

**Maintainer** Lars Otto <larsot23@gmail.com>

**Depends** lpSolveAPI, ucminf

**Imports** methods, stats, graphics, grDevices

**Description** Methods for frontier analysis, Data Envelopment Analysis (DEA), under different technology assumptions (fdh, vrs, drs, crs, irs, add/frh, and fdh+), and using different efficiency measures (input based, output based, hyperbolic graph, additive, super, and directional efficiency). Peers and slacks are available, partial price information can be included, and optimal cost, revenue and profit can be calculated. Evaluation of mergers is also supported. Methods for graphing the technology sets are also included. There is also support comparative methods based on Stochastic Frontier Analyses (SFA). In general, the methods can be used to solve not only standard models, but also many other model variants. It complements the book, Bogetoft and Otto, Benchmarking with DEA, SFA, and R, Springer-Verlag, 2011, but can of course also be used as a stand-alone package.

**License** GPL (>= 2)

**LazyLoad** yes

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2018-05-10 13:53:29 UTC

## R topics documented:

Benchmarking-package . . . . . 2

charnes1981 . . . . .	5
cost.opt . . . . .	7
critValue . . . . .	10
dea . . . . .	11
dea.add . . . . .	15
dea.boot . . . . .	17
dea.direct . . . . .	20
dea.dual . . . . .	24
dea.merge . . . . .	26
dea.plot . . . . .	28
eff, efficiencies . . . . .	31
eff.dens . . . . .	33
eladder . . . . .	34
excess . . . . .	36
lambda . . . . .	37
make.merge . . . . .	38
malmq . . . . .	40
malmquist . . . . .	42
mea . . . . .	44
milkProd . . . . .	46
norWood2004 . . . . .	47
outlier.ap . . . . .	48
peers . . . . .	49
pigdata . . . . .	51
projekt . . . . .	52
sdea . . . . .	53
sfa . . . . .	55
slack . . . . .	58
typeIerror . . . . .	60
<b>Index</b>	<b>62</b>

---

Benchmarking-package     *Data Envelopment Analyses (DEA) and Stochastic Frontier Analyses (SFA) – Model Estimations and Efficiency Measuring*

---

## Description

The Benchmarking package contains methods to estimate technologies and measure efficiencies using DEA and SFA. Data Envelopment Analysis (DEA) are supported under different technology assumptions (fdh, vrs, drs, crs, irs, add), and using different efficiency measures (input based, output based, hyperbolic graph, additive, super, directional). Peers are available, partial price information can be included, and optimal cost, revenue and profit can be calculated. Evaluation of mergers are also supported. A comparative method for estimating stochastic frontier function (SFA) efficiencies is included. The methods can solve not only standard models, but also many other model variants, and they can be modified to solve new models.

The package also support simple plots of DEA technologies with two goods; either as a transformation curve (2 outputs), an isoquant (2 inputs), or a production function (1 input and 1 output). When more inputs and outputs are available they are aggregated using weights (prices, relative prices).

The package complements the book, Bogetoft and Otto, *Benchmarking with DEA, SFA, and R*, Springer-Verlag 2011, but can of course also be used as a stand-alone package.

## Details

Package: Benchmarking  
 Type: Package  
 Version: 0.27 (\$Revision: 157 \$)  
 Date: \$Date: 2015-07-09 11:40:20 +0200 (to, 09 jul 2015) \$  
 License: Copyright

dea	DEA input or output efficiency measures, peers, lambdas and slacks
dea.dual	Dual weights (prices), including restrictions on weights
dea.direct	Directional efficiency
sdea	Super efficiency.
dea.add	Additive efficiency; sum of slacks in DEA technology.
mea	Multidirectional efficiency analysis or potential improvements.
eff	Efficiency from an object returned from any of the dea or sfa functions.
slack	Slacks in DEA models
excess	Calculates excess input or output compared to DEA frontier.
peers	get the peers for each firm.
dea.boot	Bootstrap DEA models
cost.opt	Optimal input for given output and prices.
revenue.opt	Optimal output for given input and prices.
profit.opt	Optimal input and output for given input and output prices.
dea.plot	Graphs of DEA technologies under alternative technology assumptions.
dea.plot.frontier	Specialized for 1 input and 1 output.
dea.plot.isoquant	Specialized for 2 inputs.
dea.plot.transform	Specialized for 2 outputs.
eladder	Efficiency ladder for a single firm.
eladder.plot	Plot efficiency ladder for a single firm.
make.merge	Make an aggregation matrix to perform mergers.
dea.merge	Decompose efficiency from a merger of firms
sfa	Stochastic frontier analysis, production, distance, and cost function (SFA)
outlier.ap	Detection of outliers
eff.dens	Estimate and plot kernel density of efficiencies
critValue	Critical values calculated from bootstrap DEA models.
typeIerror	Probability of a type I error for a test in bootstrap DEA models.

**Note**

The interface for the methods are very much like the interface to the methods in the package **FEAR** (Wilson 2008). One change is that the data now are transposed to reflect how data is usually available in applications, i.e. we have firms on rows, and inputs and output in the columns. Also, the argument for the options RTS and ORIENTATION can be given as memotechnical strings, and there are more options to control output.

The input and output matrices can contain negative numbers, and the methods can thereby manage restricted or fixed input or output.

The return is not just the efficiency, but also slacks, dual values (shadow prices), peers, and lambdas (weights).

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**References**

Bogetoft and Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011

Paul W. Wilson (2008), "FEAR 1.0: A Software Package for Frontier Efficiency Analysis with R," *Socio-Economic Planning Sciences* 42, 247–254

**Examples**

```
# Plot of different technologies
x <- matrix(c(100,200,300,500),ncol=1,dimnames=list(LETTERS[1:4],"x"))
y <- matrix(c(75,100,300,400),ncol=1,dimnames=list(LETTERS[1:4],"y"))
dea.plot(x,y,RTS="vrs",ORIENTATION="in-out",txt=rownames(x))
dea.plot(x,y,RTS="drs",ORIENTATION="in-out",add=TRUE,lty="dashed",lwd=2)
dea.plot(x,y,RTS="crs",ORIENTATION="in-out",add=TRUE,lty="dotted")

dea.plot(x,y,RTS="fdh",ORIENTATION="in-out",txt=rownames(x),main="fdh")
dea.plot(x,y,RTS="irs",ORIENTATION="in-out",txt=TRUE,main="irs")
dea.plot(x,y,RTS="irs2",ORIENTATION="in-out",txt=rownames(x),main="irs2")
dea.plot(x,y,RTS="add",ORIENTATION="in-out",txt=rownames(x),main="add")

# A quick frontier with 1 input and 1 output
dea.plot(x,y, main="Basic plot of frontier")

# Calculating efficiency
dea(x,y, RTS="vrs", ORIENTATION="in")
e <- dea(x,y, RTS="vrs", ORIENTATION="in")
e
eff(e)
peers(e)
peers(e, NAMES=TRUE)
print(peers(e, NAMES=TRUE), quote=FALSE)
lambda(e)
summary(e)
```

```

# Calculating super efficiency
esuper <- sdea(x,y, RTS="vrs", ORIENTATION="in")
esuper
print(peers(esuper,NAMES=TRUE),quote=FALSE)
# Technology for super efficiency for firm number 3/C
# Note that drop=FALSE is necessary for XREF and YREF to be matrices
# when one of the dimensions is or is reduced to 1.
e3 <- dea(x,y, XREF=x[-3,,drop=FALSE], YREF=y[-3,,drop=FALSE])
dea.plot(x[-3],y[-3],RTS="vrs",ORIENTATION="in-out",txt=LETTERS[c(1,2,4)])
points(x[3],y[3],cex=2)
text(x[3],y[3],LETTERS[3],adj=c(-.75,.75))
e3 <- dea(x,y, XREF=x[-3,,drop=FALSE], YREF=y[-3,,drop=FALSE])
eff(e3)
peers(e3)
print(peers(e3,NAMES=TRUE),quote=FALSE)
lambda(e3)
e3$lambda

# Taking care of slacks
x <- matrix(c(100,200,300,500,100,600),ncol=1,
            dimnames=list(LETTERS[1:6],"x"))
y <- matrix(c(75,100,300,400,50,400),ncol=1,
            dimnames=list(LETTERS[1:6],"y"))

# Phase one, calculate efficiency
e <- dea(x,y)
print(e)
peers(e)
lambda(e)
# Phase two, calculate slacks (maximize sum of slacks)
sl <- slack(x,y,e)
data.frame(sl$sx,sl$sy)
peers(sl)
lambda(sl)
sl$lambda
summary(sl)

# The two phases in one function call
e2 <- dea(x,y,SLACK=TRUE)
print(e2)
data.frame(eff(e2),e2$slack,e2$sx,e2$sy,lambda(e2))
peers(e2)
lambda(e2)
e2$lambda

```

## Description

The data set is from an US federally sponsored program for providing remedial assistance to disadvantaged primary school students. The firms are 70 school sites, and data are from entire sites. The variables consists of results from three different kind of tests, a reading score,  $y_1$ , a math score,  $y_2$ , and a self-esteem score,  $y_3$ , which are considered outputs in the model, and five different variables considered to be inputs, the education level of the mother,  $x_1$ , the highest occupation of a family member,  $x_2$ , parental visits to school,  $x_3$ , time spent with children in school-related topics,  $x_4$ , and the number of teachers at the site,  $x_5$ .

## Usage

```
data(charnes1981)
```

## Format

A data frame with 70 school sites with the following variables.

```
firm school site number
x1 education level of the mother
x2 highest occupation of a family member
x3 parental visits to school
x4 time spent with children in school-related topics
x5 the number of teachers at the site
y1 reading score
y2 math score
y3 self-esteem score
pft =1 if in program (program follow through) and =0 if not in program
name Site name
```

## Details

The command `data(charnes1981)` will create a data frame named `charnes1981` with the above data.

Beside input and output variables there is further information in the data set, that the first 50 school sites followed the program and that the last 20 are the results for sites not following the program. This is showed by the variable `pft`.

## Note

Data as `.csv` are loaded by the command `data` using `read.table(..., header=TRUE, sep=";")` such that this file is a semicolon separated file and not a comma separated file.

Therefore, to read the file from a script the command must be `read.csv("charnes1981.csv", sep=";")` or `read.csv2("charnes1981.csv")`.

Thus the data can be read either as `charnes1981 <- read.csv2(paste(.Library, "Benchmarking/data", "charnes1981.csv", sep = "/"))` or as `data(charnes1981)` if the package **Benchmarking** is loaded. In both cases the data will be in the data frame `charnes1981`.

**Source**

Charnes, Cooper, and Rhodes, “Evaluating Program and Managerial Efficiency: An Application of Data Envelopment Analysis to Program Follow Through”, *Management Science*, volume 27, number 6, June 1981, pages 668–697.

**Examples**

```
data(charnes1981)
x <- with(charnes1981, cbind(x1,x2,x3,x4,x5))
y <- with(charnes1981, cbind(y1,y2,y3))

# Farrell input efficiency; vrs technology
e <- dea(x,y)
# The number of times each peer is a peer
np <- get.number.peers(e)
# Peers that are peers for more than 20 schools, and the number of
# times they are peers
np[which(np[,2]>20),]

# Plot first input against first output and emphasize the peers that
# are peers for more than 20 schools in the model with five inputs and
# three outputs
inp <- np[which(np[,2]>20),1]
dea.plot(x[,1],y[,1])
points(x[inp,1], y[inp,1], pch=16, col="red")
```

---

cost.opt

*DEA optimal cost, revenue, and profit*


---

**Description**

Estimates the input and/or output vector(s) that minimize cost, maximize revenue or maximize profit in the context of a DEA technology

**Usage**

```
cost.opt(XREF, YREF, W, YOBS=NULL, RTS="vrs", param=NULL,
         TRANSPOSE=FALSE, LP=FALSE, LPK = NULL)
```

```
revenue.opt(XREF, YREF, P, XOBS=NULL, RTS="vrs", param=NULL,
            TRANSPOSE = FALSE, LP = FALSE, LPK = NULL)
```

```
profit.opt(XREF, YREF, W, P, RTS = "vrs", param=NULL,
           TRANSPOSE = FALSE, LP = FALSE, LPK = NULL)
```

## Arguments

		Input and output matrices are in the same form as for the method <a href="#">dea</a> .
XREF		Input of the firms defining the technology, a $K \times m$ matrix of observations of $K$ firms with $m$ inputs (firm $\times$ input). In case <code>TRANSPPOSE=TRUE</code> the input matrix is transposed as input $\times$ firm.
YREF		output of the firms defining the technology, a $K \times n$ matrix of observations of $K$ firms with $n$ outputs (firm $\times$ input). In case <code>TRANSPPOSE=TRUE</code> the output matrix is transposed as output $\times$ firm.
W		Input prices as a matrix. Either same prices for all firms or individual prices for all firms; i.e. either a $1 \times m$ or a $K \times m$ matrix for $K$ firms and $m$ inputs
P		Output prices as a matrix. Either same prices for all firms or individual prices for all firms; i.e. either a $1 \times n$ or $K \times n$ matrix for $K$ firms and $n$ outputs
XOBS		The input for which an optimal, revenue maximizing, output vector is to be calculated. Defaults is XREF. Same form as XREF
YOBS		The output for which an optimal, cost minimizing input vector is to be calculated. Defaults is YREF. Same form as YREF
RTS		A text string or a number defining the underlying DEA technology / returns to scale assumption.
0	fdh	Free disposability hull, no convexity assumption
1	vrs	Variable returns to scale, convexity and free disposability
2	drs	Decreasing returns to scale, convexity, down-scaling and free disposability
3	crs	Constant returns to scale, convexity and free disposability
4	irs	Increasing returns to scale, (up-scaling, but not down-scaling), convexity and free disposability
5	add	Additivity (scaling up and down, but only with integers), and free disposability
6	fdh+	A combination of free disposability and restricted or local constant return to scale
param		Possible parameters. At the moment only used for <code>RTS="fdh+"</code> to set low and high values for restrictions on lambda; see the section details and examples in <a href="#">dea</a> for its use. Future versions might also use <code>param</code> for other purposes.
TRANSPPOSE		Input and output matrices are treated as firms times goods for the default value <code>TRANSPPOSE=FALSE</code> corresponding to the standard in R for statistical models. When <code>TRUE</code> data matrices, quantities and prices, are transposed to goods times firms matrices.
LP		Only for debugging. If <code>LP=TRUE</code> then input and output for the LP program are written to standard output for each unit.
LPK		When <code>LPK=k</code> then a mps file is written for firm $k$ ; it can be used as input to an alternative LP solver to check the results.

## Details

The LP optimization problem is formulated in Bogetoft and Otto (2011, pp 35 and 102) and is solved by the LP method in the package **lpSolveAPI**.

The methods `print` and `summary` are working for `cost.opt`, `revenue.opt`, and `profit.opt`



**Value**

The values returned are the optimal input, and/or optimal output. When saved in an object the following components are available:

xopt	The optimal input, returned as a matrix by cost.opt and profit.cost.
yopt	The optimal output, returned as a matrix by revenue.opt and profit.cost.
cost	The optimal/minimal cost.
revenue	The optimal/maximal revenue
profit	The optimal/maximal profit
lambda	The peer weights that determines the technology, a matrix. Each row is the lambdas for the firm corresponding to that row; for the vrs technology the rows sum to 1. A column shows for a given firm how other firms are compared to this firm; i.e. peers are firms with a positive element in their columns.

**Note**

The index for peer units can be returned by the method peers and the weights are returned in lambda. Note that the peers now are the firms for the optimal input and/or output allocation, not just the technical efficient firms.

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**References**

Bogetoft and Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011

**See Also**

Paul W. Wilson (2008), "FEAR 1.0: A Software Package for Frontier Efficiency Analysis with R," *Socio-Economic Planning Sciences* 42, 247–254

**Examples**

```
x <- matrix(c(2,12, 2,8, 5,5, 10,4, 10,6, 3,13), ncol=2, byrow=TRUE)
y <- matrix(1,nrow=dim(x)[1],ncol=1)
w <- matrix(c(1.5, 1),ncol=2)

txt <- LETTERS[1:dim(x)[1]]
dea.plot(x[,1],x[,2], ORIENTATION="in", cex=1.25)
text(x[,1],x[,2],txt,adj=c(-.7,-.2),cex=1.25)

# technical efficiency
te <- dea(x,y,RTS="vrs")
xopt <- cost.opt(x,y,w,RTS=1)
cobs <- x %*% t(w)
copt <- xopt$x %*% t(w)
```

```

# cost efficiency
ce <- copt/cobs
# allocative efficiency
ae <- ce/te$eff
data.frame("ce"=ce,"te"=te$eff,"ae"=ae)
print(cbind("ce"=c(ce),"te"=te$eff,"ae"=c(ae)),digits=2)

# isocost line in the technology plot
abline(a=copt[1]/w[2], b=-w[1]/w[2], lty="dashed")

```

---

critValue

*Critical values from bootstrapped DEA models*


---

### Description

Calculates critical value for test using bootstrap output in DEA models

### Usage

```
critValue(s, alpha=0.05)
```

### Arguments

s	Vector with calculated values of the statistic for each of the NREP bootstraps; NREP is from boot.sw98
alpha	The size of the test

### Details

Needs bootstrapped values of the test statistic

### Value

Returns the critical value

### Author(s)

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

### See Also

boot.sw98 in **FEAR**, Paul W. Wilson (2008), "FEAR 1.0: A Software Package for Frontier Efficiency Analysis with R," *Socio-Economic Planning Sciences* 42, 247–254

### Examples

```

# The critical value for two-sided test in normal distribution found
# by simulation.
x <- rnorm(1000000)
critValue(x,.975)

```

---

dea *DEA efficiency*

---

### Description

Estimates a DEA frontier and calculates efficiency measures a la Farrell.

### Usage

```
dea(X, Y, RTS="vrs", ORIENTATION="in", XREF=NULL, YREF=NULL,
    FRONT.IDX=NULL, SLACK=FALSE, DUAL=FALSE, DIRECT=NULL, param=NULL,
    TRANSPPOSE=FALSE, FAST=FALSE, LP=FALSE, CONTROL=NULL, LPK=NULL)
```

```
## S3 method for class 'Farrell'
print(x, digits=4, ...)
## S3 method for class 'Farrell'
summary(object, digits=4, ...)
```

### Arguments

X	Inputs of firms to be evaluated, a K x m matrix of observations of K firms with m inputs (firm x input). In case TRANSPPOSE=TRUE the input matrix is transposed to input x firm.
Y	Outputs of firms to be evaluated, a K x n matrix of observations of K firms with n outputs (firm x input). In case TRANSPPOSE=TRUE the output matrix is transposed to output x firm.
RTS	Text string or a number defining the underlying DEA technology / returns to scale assumption.
0 fdh	Free disposability hull, no convexity assumption
1 vrs	Variable returns to scale, convexity and free disposability
2 drs	Decreasing returns to scale, convexity, down-scaling and free disposability
3 crs	Constant returns to scale, convexity and free disposability
4 irs	Increasing returns to scale, (up-scaling, but not down-scaling), convexity and free disposability
5 irs2	Increasing returns to scale (up-scaling, but not down-scaling), additivity, and free disposability
6 add	Additivity (scaling up and down, but only with integers), and free disposability; also known as replicability and free disposability
7 fdh+	A combination of free disposability and restricted or local constant return to scale
10 vrs+	As vrs, but with restrictions on the individual lambdas via param
ORIENTATION	Input efficiency "in" (1), output efficiency "out" (2), and graph efficiency "graph" (3). For use with DIRECT, an additional option is "in-out" (0).
XREF	Inputs of the firms determining the technology, defaults to X

YREF	Outputs of the firms determining the technology, defaults to Y
FRONT .IDX	Index for firms determining the technology
SLACK	Calculate slack in a phase II calculation by an intern call of the function <code>slack</code> . Note that the precision for calculating slacks for orientation graph is low.
DUAL	Calculate dual variables, i.e. shadow prices; not calculated for orientation graph as that is not an LP problem.
DIRECT	Directional efficiency, DIRECT is either a scalar, an array, or a matrix with non-negative elements. If the argument is a scalar, the direction is (1,1,...,1) times the scalar; the value of the efficiency depends on the scalar as well as on the unit of measurements. If the argument is an array, this is used for the direction for every firm; the length of the array must correspond to the number of inputs and/or outputs depending on the ORIENTATION. If the argument is a matrix then different directions are used for each firm. The dimensions depends on the ORIENTATION (and TRANSPOSE), the number of firms must correspond to the number of firms in X and Y. DIRECT must not be used in connection with DIRECTION="graph".
param	Possible parameters. At the moment only used for RTS="fdh+" to set low and high values for restrictions on lambda; see the section details and examples for its use. Future versions might also use param for other purposes.
TRANSPOSE	Input and output matrices are treated as firms times goods matrices for the default value TRANSPOSE=FALSE corresponding to the standard in R for statistical models. When TRUE data matrices are transposed to good times firms matrices as is normally used in LP formulation of the problem.
LP	Only for debugging. If LP=TRUE then input and output for the LP program are written to standard output for each unit.
FAST	Only calculate efficiencies and just return them as a vector, i.e. no lambda or other output. The return when using FAST cannot be used as input for <code>slack</code> and <code>peers</code> .
CONTROL	Possible controls to <b>lpSolveAPI</b> , see the documentation for that package.
...	Optional parameters for the print and summary methods.
object, x	An object of class Farrell (returned by the function <code>dea</code> ) – R code uses ‘object’ and ‘x’ alternating for generic methods.
digits	digits in printed output, handled by format in print.
LPK	when LPK=k then a mps file is written for firm k; it can be used as input to an alternative LP solver to check the results.

## Details

The return from `dea` and `sdea` is an object of class Farrell. The efficiency in `dea` is calculated by the LP method in the package **lpSolveAPI**. Slacks can be calculated either in the call of `dea` using the option `SLACK=TRUE` or in a following call to the function `slack`.

The directional efficiency when the argument `DIRECT` is used, depends on the unit of measurement and is not restricted to be less than 1 (or greater than 1 for output efficiency) and is therefore completely different from the Farrell efficiency.

The crs factor in `RTS="fdh+"` that sets the lower and upper bound can be changed by the argument `param` that will set the lower and upper bound to  $1-\text{param}$  and  $1+\text{param}$ ; the default value is `param=.15`. The value must be greater than or equal to 0 and strictly less than 1. A value of 0 corresponds to `RTS="fdh"`. To get an asymmetric interval set `param` to a 2 dimensional array with values for the low and high end for interval, for instance `param=c(.8,1.15)`. The FDH+ technology set is described in Bogetoft and Otto (2011) pages 73–74.

The technology `RTS="vrs+"` uses the parameter `param` to set restrictions on `lambda`, the convexity parameters. The elements of `param` are `param=(low, high, sum_low, sum_high)` where "low" and "high" are restrictions on the individual `lambda` and "sum\_low" and "sum\_high" are restrictions on the sum of `lambdas`. The individual `lambda` must be in the interval from low to high or be zero. With one parameter the restrictions set are `(param, 1+1-(param), 1, 1)`, with two parameters `(param[1], param[2], 1, 1)`, and with four parameters `(param[1], param[2], param[3], param[4])`. The resulting technology set is not necessarily convex.

The graph orientated efficiency is calculated by bisection between feasible and infeasible values of `G`. The precision in the result is less than for the other orientations.

When the argument `DIRECT=d` is used then the returned value `e` for input orientation is the excess input measured in `d` units of measurements, i.e.  $x - ed$ , and for output orientation  $y + ed$ . The directional efficiency can be restricted to inputs (`ORIENTATION="in"`), restricted to outputs (`ORIENTATION="out"`), or both include inputs and output directions (`ORIENTATION="in-out"`). Directional efficiency is discussed on pages 31–35 and 121–127 in Bogetoft and Otto (2011).

## Value

The results are returned in a Farrell object with the following components. The last three components in the list are only part of the object when `SLACK=TRUE`.

<code>eff</code>	The efficiencies. Note when <code>DIRECT</code> is used then the efficiencies are not Farrell efficiencies but rather excess values in <code>DIRECT</code> units of measurement
<code>lambda</code>	The <code>lambdas</code> , i.e. the weight of the peers, for each firm
<code>objval</code>	The objective value as returned from the LP program; normally the same as <code>eff</code> , but for <code>slack</code> it is the the sum of the slacks.
<code>RTS</code>	The return to scale assumption as in the option <code>RTS</code> in the call
<code>ORIENTATION</code>	The efficiency orientation as in the call
<code>TRANSPOSE</code>	As in the call
<code>slack</code>	A logical vector where the component for a firm is <code>TRUE</code> if the sums of slacks for the corresponding firm is positive. Only calculated in <code>dea</code> when option <code>SLACK=TRUE</code>
<code>sum</code>	A vector with sums of the slacks for each firm. Only calculated in <code>dea</code> when option <code>SLACK=TRUE</code>
<code>sx</code>	A matrix for input slacks for each firm, only calculated if the option <code>SLACK</code> is <code>TRUE</code> or returned from the method <code>slack</code>
<code>sy</code>	A matrix for output slack, see <code>sx</code>
<code>ux</code>	Dual variable for input, only calculated if <code>DUAL</code> is <code>TRUE</code> .
<code>vy</code>	Dual variable for output, only calculated if <code>DUAL</code> is <code>TRUE</code> .

**Note**

The arguments X, Y, XREF, and YREF are supposed to be matrices or numerical data frames that in the function will be converted to matrices. When subsetting a matrix or data frame to just one column then the class of the resulting object/variable is no longer a matrix or a data frame, but just a numeric (array, vector). Therefore, in this case a numeric input that is not a matrix nor a data frame is transformed to a 1 column matrix, and here the use of the argument TRANSPOSE=TRUE gives an error.

The dual values are not unique for extreme points (firms on the boundary with an efficiency of 1) and therefore the calculated dual values for these firms can depend on the order of firms in the reference technology. The same lack of uniqueness also make the peers for some firms depend on the order of firms in the reference technology.

To calculate slack use the argument SLACK=TRUE or use the function `slack` directly.

When there is slack, and slack is not taken into consideration, then the peers for a firm with slack might depend on the order of firms in the data set; this is a property of the LP algorithm used to solve the problem.

To handle fixed, non-discretionary inputs, one can let it appear as negative output in an input-based mode, and reversely for fixed, non-discretionary outputs. Fixed inputs (outputs) can also be handled by directional efficiency; set the direction, the argument DIRECT, equal to the variable, discretionary inputs (outputs) and 0 for the fixed inputs (outputs).

When the the argument DIRECT=X is used the then the returned efficiency is equal to 1 minus the Farrell efficiency for input orientation and to the Farrell efficiency minus 1 for output orientation.

To use matrices X and Y prepared for the methods in the package **FEAR** (Wilson 2008) set the options TRANSPOSE=TRUE; for consistency with **FEAR** the options RTS and ORIENTATION also accepts numbers as in **FEAR**.

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**References**

Peter Bogetoft and Lars Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011

**See Also**

Paul W. Wilson (2008), "FEAR 1.0: A Software Package for Frontier Efficiency Analysis with R," *Socio-Economic Planning Sciences* 42, 247–254

**Examples**

```
x <- matrix(c(100,200,300,500,100,200,600),ncol=1)
y <- matrix(c(75,100,300,400,25,50,400),ncol=1)
dea.plot.frontier(x,y,txt=TRUE)

e <- dea(x,y)
eff(e)
print(e)
summary(e)
```

```

lambda(e)

# Input savings potential for each firm
(1-eff(e)) * x
(1-e$eff) * x

# calculate slacks
el <- dea(x,y,SLACK=TRUE)
data.frame(e$eff,el$eff,el$slack,el$sx,el$sy)

# Fully efficient units, eff==1 and no slack
which(eff(e) == 1 & !el$slack)

# fdh+ with limits in the interval [.7, 1.2]
dea(x,y,RTS="fdh+", param=c(.7,1.2))

```

---

 dea.add

*Additive DEA model*


---

### Description

Calculates additive efficiency as sum of input and output slacks within different DEA models

### Usage

```

dea.add(X, Y, RTS="vrs", XREF=NULL, YREF=NULL,
        FRONT.IDX=NULL, param=NULL, TRANSPPOSE=FALSE, LP=FALSE)

```

### Arguments

X	Inputs of firms to be evaluated, a K x m matrix of observations of K firms with m inputs (firm x input). In case TRANSPPOSE=TRUE the input matrix is transposed to input x firm.
Y	Outputs of firms to be evaluated, a K x n matrix of observations of K firms with n outputs (firm x input). In case TRANSPPOSE=TRUE the output matrix is transposed to output x firm.
RTS	Text string or a number defining the underlying DEA technology / returns to scale assumption.
0	fdh Free disposability hull, no convexity assumption
1	vrs Variable returns to scale, convexity and free disposability
2	drs Decreasing returns to scale, convexity, down-scaling and free disposability
3	crs Constant returns to scale, convexity and free disposability
4	irs Increasing returns to scale, (up-scaling, but not down-scaling), convexity and free disposability
5	add Additivity (scaling up and down, but only with integers), and free disposability

XREF	Inputs of the firms determining the technology, defaults to X
YREF	Outputs of the firms determining the technology, defaults to Y
FRONT .IDX	Index for firms determining the technology
param	Possible parameters. At the moment only used for RTS="fdh+" to set low and high values for restrictions on lambda; see the section details and examples for its use. Future versions might also use param for other purposes.
TRANSDPOSE	Input and output matrices are treated as firms times goods matrices for the default value TRANSDPOSE=FALSE corresponding to the standard in R for statistical models. When TRUE data matrices are transposed to good times firms matrices as is normally used in LP formulation of the problem.
LP	Only for debugging. If LP=TRUE then input and output for the LP program are written to standard output for each unit.

### Details

The sum of the slacks is maximized in a LP formulation of the DEA technology. The sum of the slacks can be seen as distance to the frontier when you only move parallel to the axes of inputs and outputs, i.e. not a usual Euclidean distance, but what is also known as an L1 norm.

Since it is the sum of slacks that is calculated, there is no exogenous ORIENTATION in the problem. Rather, there is generally both an input and an output direction in the slacks. The model considers the input excess and output shortfall simultaneously and finds a point on the frontier that is most distant to the point being evaluated.

### Value

sum	Sum of all slacks for each firm, $sum = sum(sx) + sum(sy)$ .
slack	A non-NULL vector of logical variables, TRUE if there is slack for the corresponding firm, and FALSE if there is no slack, i.e. the sum of slacks is zero.
sx	A matrix of input slacks for each firm
sy	A matrix of output slack for each firm
lambda	The lambdas, i.e. the weights of the peers for each firm

### Note

This is neither a Farrell nor a Shephard like efficiency.

The value of the slacks depends on the scaling of the different inputs and outputs. Therefore the values are not independent of how the input and output are measured.

### Author(s)

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

### Source

Corresponds to Eqs. 4.34-4.38 in Cooper et al. (2007)



## References

Bogetoft and Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011

Cooper, Seiford, and Tone; *Data Envelopment Analysis: A Comprehensive Text with Models, Applications, References and DEA-Solver Software*; Second edition, Springer 2007

## Examples

```
x <- matrix(c(2,3,2,4,6,5,6,8),ncol=1)
y <- matrix(c(1,3,2,3,5,2,3,5),ncol=1)
dea.plot.frontier(x,y,txt=1:dim(x)[1])

sb <- dea.add(x,y,RTS="vrs")
data.frame("sx"=sb$sx,"sy"=sb$sy,"sum"=sb$sum,"slack"=sb$slack)
```

---

dea.boot

*Bootstrap DEA models*

---

## Description

The function `dea.boot` bootstrap DEA models and returns bootstrap of Farrell efficiencies. This function is slower than the `boot.sw89` from the package **FEAR**. The faster function `boot.fear` is a wrapper for `boot.sw89` from the package **FEAR** returning results directly as Farrell measures.

## Usage

```
dea.boot(X, Y, NREP = 200, EFF = NULL, RTS = "vrs", ORIENTATION="in",
         alpha = 0.05, XREF = NULL, YREF = NULL, EREF = NULL,
         DIRECT = NULL, TRANSPOSE = FALSE, SHEPHARD.INPUT = TRUE, LP)

boot.fear(X, Y, NREP = 200, EFF = NULL, RTS = "vrs", ORIENTATION = "in",
          alpha = 0.05, XREF = NULL, YREF = NULL, EREF = NULL)
```

## Arguments

X	Inputs of firms to be evaluated, a K x m matrix of observations of K firms with m inputs (firm x input)
Y	Outputs of firms to be evaluated, a K x n matrix of observations of K firms with n outputs (firm x input).
NREP	Number of bootstrap replicats
EFF	Efficiencies for (X,Y) relative to the technology generated from (XREF,YREF).
RTS	The returns to scale assumptions as in <a href="#">dea</a> , only works for "vrs", "drs", and "crs"; more to come.
ORIENTATION	Input efficiency "in" (1), output efficiency "out" (2), and graph efficiency "graph" (3).
alpha	One minus the size of the confidence interval for the bias corrected efficiencies

XREF	Inputs of the firms determining the technology, defaults to X.
YREF	Outputs of the firms determining the technology, defaults to Y.
EREF	Efficiencies for the firms in XREF, YREF.
DIRECT	Does not yet work and is therefore not used.
TRANSPPOSE	Input and output matrices are $K \times m$ and $K \times n$ for the default value <code>TRANSPPOSE=FALSE</code> ; this is standard in R for statistical models. When <code>TRANSPPOSE=TRUE</code> data matrices are $m \times K$ and $n \times K$ .
SHEPHARD.INPUT	The bootstrap of the Farrell input efficiencies is done as a Shephard input distance function, the inverse Farrell input efficiency. The option is only relevant for input and graph directions.
LP	Only for debugging purposes.

### Details

The details are lightly explained in Bogetoft and Otto (2011) Chap. 6, and with more mathematical details in Dario and Simar (2007) Sect. 3.4 and in Simar and Wilson (1998).

The bootstrap at the moment does not work for any kind of directional efficiency.

The returned confidence intervals are for the bias corrected efficiencies; to get confidence intervals for the uncorrected efficiencies add the biases to both upper and lower values for the intervals.

Under the default option `SHEPHARD.INPUT=TRUE` bias and bias corrected efficiencies are calculated for Shephard input distance function and then transformed to Farrell input efficiencies to avoid possible negative biased corrected input efficiencies. If this is not wanted use the option `SHEPHARD.INPUT=FALSE`. This option is only relevant for input and graph oriented directions.

### Value

The returned values from both functions are as follows:

eff	Efficiencies
eff.bc	Bias-corrected efficiencies
bias	An array of bootstrap bias estimates for the K firms
conf.int	$K \times 2$ matrix with confidence interval for the estimated efficiencies
var	An array of bootstrap variance estimates for the K firms
boot	The replica bootstrap estimates of the Farrell efficiencies, a $K \times NREP$ matrix

### Note

The function `dea.boot` does not depend on the FEAR package and can therefore be used on computers where the package FEAR is not available. This, however, comes with a time penalty as it takes around 4 times longer to run compared to using FEAR directly

The returned bootstrap estimates from `FEAR::boot.sw98` of efficiencies are sorted for each firm individually.

Unfortunately, this means that the component of replicas is not the efficiencies for the same bootstrap replica, but could easily be from different bootstrap replicas. This also means that this function can *not* be used to bootstrap tests for statistical hypotheses where the statistics involves summing of firms efficiencies.

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**References**

Peter Bogetoft and Lars Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011.

Cinzia Dario and L. Simar; *Advanced Robust and Nonparametric Methods in Efficiency Analysis. Methodology and Applications*; Springer 2007.

Leopold Simar and Paul .W. Wilson (1998), “Sensitivity analysis of efficiency scores: How to bootstrap in nonparametric frontier models”, *Management Science* 44, 49–61.

Paul W. Wilson (2008), “FEAR 1.0: A Software Package for Frontier Efficiency Analysis with R,” *Socio-Economic Planning Sciences* 42, 247–254

**See Also**

The documentation for boot . sw98 in the package **FEAR**.

**Examples**

```
x <- matrix(c(100,200,300,500,100,200,600),ncol=1)
y <- matrix(c(75,100,300,400,25,50,400),ncol=1)

e <- dea(x,y)
eff(e)

dea.plot.frontier(x,y,txt=TRUE)

# To bootstrap for real, NREP should be at least 2000. Run the
# following lines a couple of times with nrep=100 and see how the
# bootstrap frontier changes from one run to the next. Try the same
# with NREP=2000 even though it does take a longer time to run,
# especially for dea.boot.
nrep <- 1
# nrep <- 2000

# if ( "FEAR" %in% .packages(TRUE) ) {
## The following only works if the package FEAR is installed; it does
## not have to be loaded.
# b <- boot.fear(x,y, NREP=nrep)
# } else {
  b <- dea.boot(x,y, NREP=nrep)
# }

# bias corrected frontier
dea.plot.frontier(b$eff.bc*x, y, add=TRUE, lty="dashed")
# outer 95% confidence interval frontier for uncorrected frontier
dea.plot.frontier((b$conf.int[,1]+b$bias)*x, y, add=TRUE, lty="dotted")

## Test of hypothesis in DEA model
```

```

# Null hypothesis is that technology is CRS and the alternative is VRS
# Bogetoft and Otto (2011) pages 183--185.
ec <- dea(x,y, RTS="crs")
Ec <- eff(ec)
ev <- dea(x,y, RTS="vrs")
Ev <- eff(ev)
# The test statistic; equation (6.1)
S <- sum(Ec)/sum(Ev)

# To calculate CRS and VRS efficiencies in the same bootstrap replicas
# we reset the random number generator before each call of the
# function dea.boot.

# To get the an initial value for the random number generating process
# we save its state (seed)
save.seed <- sample.int(1e9,1)

# The bootstrap and calculate CRS and VRS under the assumption that
# the true technology is CRS (the null hypothesis) and such that the
# results correponds to the case where CRS and VRS are calculated for
# the same reference set of firms; to make this happen we set the
# random number generator to the same state before the calls.
set.seed(save.seed)
bc <- dea.boot(x,y, nrep,, RTS="crs")
set.seed(save.seed)
bv <- dea.boot(x,y, nrep,, RTS="vrs", XREF=x,YREF=y, EREF=ec$eff)

# Calculate the statistic for each bootstrap replica
bs <- colSums(bc$boot)/colSums(bv$boot)
# The critical value for the test (default size \code{\alpha} of test is 5%)
critValue(bs, alpha=.1)
S
# Accept the hypothesis at 10% level?
critValue(bs, alpha=.1) <= S

# The probability of observing a smaller value of S when the
# hypothesis is true; the p--value.
typeIError(S, bs)
# Accept the hypothesis at size level 10%?
typeIError(S, bs) >= .10

```

---

dea.direct

*Directional efficiency*


---

### Description

Directional efficiency rescaled to an interpretation a la Farrell efficiency and the corresponding peer importance ( $\lambda$ ).

**Usage**

```
dea.direct(X, Y, DIRECT, RTS = "vrs", ORIENTATION = "in",
           XREF = NULL, YREF = NULL, FRONT.IDX = NULL,
           SLACK = FALSE, param=NULL, TRANSPOSE = FALSE)
```

**Arguments**

X	Inputs of firms to be evaluated, a $K \times m$ matrix of observations of $K$ firms with $m$ inputs (firm $x$ input)
Y	Outputs of firms to be evaluated, a $K \times n$ matrix of observations of $K$ firms with $n$ outputs (firm $x$ input).
DIRECT	Directional efficiency, DIRECT is either a scalar, an array, or a matrix with non-negative elements.  If the argument is a scalar, the direction is $(1,1,\dots,1)$ times the scalar; the value of the efficiency depends on the scalar as well as on the unit of measurements.  If the argument an array, this is used for the direction for every firm; the length of the array must correspond to the number of inputs and/or outputs depending on the ORIENTATION.  If the argument is a matrix then different directions are used for each firm. The dimensions depends on the ORIENTATION (and TRANSPOSE), the number of firms must correspond to the number of firms in $X$ and $Y$ .  DIRECT must not be used in connection with DIRECTION="graph".
RTS	Text string or a number defining the underlying DEA technology / returns to scale assumption.
0 fdh	Free disposability hull, no convexity assumption
1 vrs	Variable returns to scale, convexity and free disposability
2 drs	Decreasing returns to scale (down-scaling, but not up-scaling), convexity, and free disposability
3 crs	Constant returns to scale, convexity and free disposability
4 irs	Increasing returns to scale (up-scaling, but not down-scaling), convexity, and free disposability
6 add	Additivity (scaling up and down, but only with integers), and free disposability
7 fdh+	A combination of free disposability and restricted or local constant return to scale
ORIENTATION	Input efficiency "in" (1), output efficiency "out" (2), and graph efficiency "graph" (3). For use with DIRECT, an additional option is "in-out" (0).
XREF	Inputs of the firms determining the technology, defaults to $X$ .
YREF	Outputs of the firms determining the technology, defaults to $Y$ .
FRONT.IDX	Index for firms determining the technology.
SLACK	See <a href="#">dea</a> and <a href="#">slack</a> .
param	Possible parameters. At the moment only used for RTS="fdh+" to set low and high values for restrictions on lambda; see the section details and examples in <a href="#">dea</a> for its use. Future versions might also use param for other purposes.
TRANSPOSE	see <a href="#">dea</a>

## Details

When the argument `DIRECT=d` is used then component `objval` of the returned object for input orientation is the maximum value of  $e$  where for input orientation  $x - ed$ , and for output orientation  $y + ed$  are in the generated technology set. The returned component `eff` is for input  $1 - ed/X$  and for output  $1 + ed/Y$  to make the interpretation as for a Farrell efficiency. Note that when the direction is not proportional to  $X$  or  $Y$  the returned `eff` are different for different inputs or outputs and `eff` is a matrix and not just an array. The directional efficiency can be restricted to inputs (`ORIENTATION="in"`), restricted to outputs (`ORIENTATION="out"`), or both include inputs and output directions (`ORIENTATION="in-out"`). Directional efficiency is discussed on pages 31–35 and 121–127 in Bogetoft and Otto (2011).

The Farrell efficiency interpretation is the ratio by which a firm can proportionally reduce all inputs (or expand all outputs) without producing less outputs (using more inputs). The directional efficiencies have the same interpretation except that the direction is not proportional to the inputs (or outputs) and therefore the different inputs may have different reduction ratios, the efficiency is an array and not just a number.

## Value

The results are returned in a Farrell object with the following components. The method `slack` only returns the three components in the list relevant for slacks.

<code>eff</code>	The Farrell efficiencies. Note that the the efficiencies are calculated to have the same interpretations as Farrell efficiencies. <code>eff</code> is a matrix if there are more than 1 good.
<code>lambda</code>	The lambdas, i.e. the weight of the peers, for each firm
<code>objval</code>	The objective value as returned from the LP program; the <code>objval</code> are excess values in <code>DIRECT</code> units of measurement.
<code>RTS</code>	The return to scale assumption as in the option <code>RTS</code> in the call
<code>ORIENTATION</code>	The efficiency orientation as in the call
<code>TRANSPOSE</code>	As in the call
<code>slack</code>	A vector with sums of the slacks for each firm. Only calculated in <code>dea</code> when option <code>SLACK=TRUE</code>
<code>sx</code>	A matrix for input slacks for each firm, only calculated if the option <code>SLACK</code> is <code>TRUE</code> or returned from the method <code>slack</code>
<code>sy</code>	A matrix for output slack, see <code>sx</code>

## Note

To handle fixed, non-discretionary inputs, one can let it appear as negative output in an input-based mode, and reversely for fixed, non-discretionary outputs. Fixed inputs (outputs) can also be handled by directional efficiency; set the direction, the argument `DIRECT`, equal to the variable, discretionary inputs (outputs) and 0 for the fixed inputs (outputs).

When the the argument `DIRECT=X` is used the then the returned efficiency is equal to 1 minus the Farrell efficiency for input orientation and equal to the Farrell efficiency minus 1 for output orientation.

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**References**

Directional efficiency is discussed on pages 31–35 and 121–127 in Bogetoft and Otto (2011).  
Peter Bogetoft and Lars Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011

**See Also**

[dea](#)

**Examples**

```
# Directional efficiency
x <- matrix(c(2,5 , 1,2 , 2,2 , 3,2 , 3,1 , 4,1), ncol=2,byrow=TRUE)
y <- matrix(1,nrow=dim(x)[1])
dea.plot.isoquant(x[,1], x[,2],txt=1:dim(x)[1])

E <- dea(x,y)
z <- c(1,1)
e <- dea.direct(x,y,DIRECT=z)
data.frame(Farrell=E$eff, Perform=e$eff, objval=e$objval)
# The direction
arrows(x[,1], x[,2], (x-z)[,1], (x-z)[,2], lty="dashed")
# The efficiency (e$objval) along the direction
segments(x[,1], x[,2], (x-e$objval*z)[,1], (x-e$objval*z)[,2], lwd=2)

# Different directions
x1 <- c(.5, 1, 2, 4, 3, 1)
x2 <- c(4, 2, 1,.5, 2, 4)
x <- cbind(x1,x2)
y <- matrix(1,nrow=dim(x)[1])
dir1 <- c(1,.25)
dir2 <- c(.25, 4)
dir3 <- c(1,4)
e <- dea(x,y)
e1 <- dea.direct(x,y,DIRECT=dir1)
e2 <- dea.direct(x,y,DIRECT=dir2)
e3 <- dea.direct(x,y,DIRECT=dir3)
data.frame(e=eff(e),e1=e1$eff,e2=e2$eff,e3=e3$eff)[6,]

# Technology and directions for all firms
dea.plot.isoquant(x[,1], x[,2],txt=1:dim(x)[1])
arrows(x[,1], x[,2], x[,1]-dir1[1], x[,2]-dir1[2],lty="dashed")
segments(x[,1], x[,2],
         x[,1]-e1$objval*dir1[1], x[,2]-e1$objval*dir1[2],lwd=2)
# slack for direction 1
dsl1 <- slack(x,y,e1)
```

```

cbind(E=e$eff,e1$eff,dsl1$sx,dsl1$sy, sum=dsl1$sum)

# Technology and directions for firm 6,
# Figure 2.6 page 32 in Bogetoft & Otto (2011)
dea.plot.isoquant(x1,x2,lwd=1.5, txt=TRUE)
arrows(x[6,1], x[6,2], x[6,1]-dir1[1], x[6,2]-dir1[2],lty="dashed")
arrows(x[6,1], x[6,2], x[6,1]-dir2[1], x[6,2]-dir2[2],lty="dashed")
arrows(x[6,1], x[6,2], x[6,1]-dir3[1], x[6,2]-dir3[2],lty="dashed")
segments(x[6,1], x[6,2],
         x[6,1]-e1$objval[6]*dir1[1], x[6,2]-e1$objval[6]*dir1[2],lwd=2)
segments(x[6,1], x[6,2],
         x[6,1]-e2$objval[6]*dir2[1], x[6,2]-e2$objval[6]*dir2[2],lwd=2)
segments(x[6,1], x[6,2],
         x[6,1]-e3$objval[6]*dir3[1], x[6,2]-e3$objval[6]*dir3[2],lwd=2)

```

---

 dea.dual

*Dual DEA models and assurance regions*


---

### Description

Solution of dual DEA models, possibly with partial value information given as restrictions on the ratios (assurance regions)

### Usage

```

dea.dual(X, Y, RTS = "vrs", ORIENTATION = "in",
        XREF = NULL, YREF = NULL,
        FRONT.IDX = NULL, DUAL = NULL, DIRECT=NULL,
        TRANSPOSE = FALSE, LP = FALSE, CONTROL=NULL, LPK=NULL)

```

### Arguments

X	Inputs of firms to be evaluated, a $K \times m$ matrix of observations of $K$ firms with $m$ inputs (firm $x$ input). In case <code>TRANSPOSE=TRUE</code> the input matrix is transposed to input $x$ firm.
Y	Outputs of firms to be evaluated, a $K \times n$ matrix of observations of $K$ firms with $n$ outputs (firm $x$ input). In case <code>TRANSPOSE=TRUE</code> the output matrix is transposed to output $x$ firm.
RTS	A text string or a number defining the underlying DEA technology / returns to scale assumption.
1	vrs Variable returns to scale, convexity and free disposability
2	drs Decreasing returns to scale, convexity, down-scaling and free disposability
3	crs Constant returns to scale, convexity and free disposability
4	irs Increasing returns to scale, (up-scaling, but not down-scaling), convexity and free disposability.



ORIENTATION	Input efficiency "in" (1), output efficiency "out" (2), and graph efficiency "graph" (3) (not yet implemented). For use with DIRECT an additional option is "in-out" (0). In this case, "graph" is not feasible
XREF	Input of the firms determining the technology, defaults to X
YREF	Output of the firms determining the technology, defaults to Y
FRONT .IDX	Index for firms determining the technology
DUAL	Matrix of order “number of input plus number of outputs minus 2” times 2. The first column is the lower bound and the second column is the upper bound for the restrictions on the multiplier ratios. The ratios are relative to the first input and the first output, respectively. This implies that there is no restriction for neither the first input nor the first output so that the number of restrictions is two less than the total number of inputs and outputs.
DIRECT	Directional efficiency, DIRECT is either a scalar, an array, or a matrix with non-negative elements. NB <i>Not yet implemented</i>
TRANSPOSE	Input and output matrices are treated as firms times goods for the default value TRANSPOSE=FALSE corresponding to the standard in R for statistical models. When TRUE data matrices shall be transposed to good times firms matrices as is normally used in LP formulation of the problem.
LP	Only for debugging. If LP=TRUE then input and output for the LP program are written to standard output for each unit.
CONTROL	Possible controls to lpSolveAPI, see the documentation for that package.
LPK	When LPK=k then a mps file is written for firm k; it can be used as input to an alternative LP solver just to check the our results.

### Details

Solved as an LP program using the package lpSolveAPI. The method `dea.dual.dea` calls the method `dea` with the option `DUAL=TRUE`.

### Value

<code>eff</code>	The efficiencies
<code>objval</code>	The objective value as returned from the LP problem, normally the same as <code>eff</code>
<code>RTS</code>	The return to scale assumption as in the option <code>RTS</code> in the call
<code>ORIENTATION</code>	The efficiency orientation as in the call
<code>TRANSPOSE</code>	As in the call
<code>u</code>	Dual values, prices, for inputs
<code>v</code>	Dual values, prices, for outputs
<code>gamma</code>	The values of gamma, the shadow price(s) for returns to scale restriction
<code>sol</code>	Solution of all variables as one component, <code>sol=c(u,v,gamma)</code> .

**Note**

Note that the dual values are not unique for extreme points in the technology set. In this case the value of the calculated dual variable can depend on the order of the complete efficient firms.

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**References**

Bogetoft and Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011. Sect. 5.10: Partial value information

**See Also**

[dea](#)

**Examples**

```
x <- matrix(c(2,5 , 1,2 , 2,2 , 3,2 , 3,1 , 4,1), ncol=2,byrow=TRUE)
y <- matrix(1,nrow=dim(x)[1])
dea.plot.isoquant(x[,1],x[,2],txt=1:dim(x)[1])
segments(0,0, x[,1], x[,2], lty="dotted")

e <- dea(x,y,RTS="crs",SLACK=TRUE)
ed <- dea.dual(x,y,RTS="crs")
print(cbind("e"=e$eff,"ed"=ed$eff, peers(e), lambda(e),
           e$sx, e$sy, ed$u, ed$v), digits=3)

dual <- matrix(c(.5, 2.5), nrow=dim(x)[2]+dim(y)[2]-2, ncol=2, byrow=TRUE)
er <- dea.dual(x,y,RTS="crs", DUAL=dual)
print(cbind("e"=e$eff,"ar"=er$eff, lambda(e), e$sx, e$sy, er$u,
           "ratio"=er$u[,2]/er$u[,1],er$v),digits=3)
```

---

dea.merge

*Estimate potential merger gains and their decompositions*

---

**Description**

Calculate and decompose potential gains from mergers of similar firms (horizontal integration).

**Usage**

```
dea.merge(X, Y, M, RTS = "vrs", ORIENTATION = "in",
          XREF = NULL, YREF = NULL, FRONT.IDX = NULL, TRANSPOSE=FALSE)
```

**Arguments**

	Most of the arguments correspond to the arguments in <a href="#">dea</a> , with $K$ firms, $m$ inputs, and $n$ outputs.
X	$K$ times $m$ matrix as in <a href="#">dea</a>
Y	$K$ times $n$ matrix as in <a href="#">dea</a>
M	$K_g$ times $K$ matrix where each row defines a merger by the firms (columns) included; matrix as returned from method <a href="#">make.merge</a>
RTS	as in <a href="#">dea</a>
ORIENTATION	as in <a href="#">dea</a>
XREF	as in <a href="#">dea</a>
YREF	as in <a href="#">dea</a>
FRONT.IDX	as in <a href="#">dea</a>
TRANSDPOSE	as in <a href="#">dea</a>

**Details**

The  $K$  firms are merged into  $K_g$  new, merged firms.

The decomposition is summarized on page 275 and in table 9.1 page 276 in Bogetoft and Otto (2011) and is based on Bogetoft and Wang (2005)

**Value**

Eff	Overall efficiencies of mergers, $K_g$ vector
Estar	Adjusted overall efficiencies of mergers after the removal of individual learning, $K_g$ vector
learning	Learning effects, $K_g$ vector
harmony	Harmony (scope) effects, $K_g$ vector
size	Size (scale) effects, $K_g$ vector

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**References**

Bogetoft and Otto; *Benchmarking with DEA, SFA, and R*; chapter 9; Springer 2011.

Bogetoft and Wang; "Estimating the Potential Gains from Mergers"; *Journal of Productivity Analysis*, 23, pp. 145-171, 2005.

**See Also**

[dea](#) and [make.merge](#)

## Examples

```
x <- matrix(c(100,200,300,500),ncol=1,dimnames=list(LETTERS[1:4],"x"))
y <- matrix(c(75,100,300,400),ncol=1,dimnames=list(LETTERS[1:4], "y"))

dea.plot.frontier(x,y,RTS="vrs",txt=LETTERS[1:length(x)],
xlim=c(0,1000),ylim=c(0,1000) )
dea.plot.frontier(x,y,RTS="drs", add=TRUE, lty="dashed", lwd=2)
dea.plot.frontier(x,y,RTS="crs", add=TRUE, lty="dotted")

dea(x,y,RTS="crs")
M <- make.merge(list(c(1,2), c(3,4)), X=x)
xmer <- M %*% x
ymer <- M %*% y
points(xmer,ymer,pch=8)
text(xmer,ymer,labels=c("A+B","C+D"),pos=4)
dea.merge(x,y,M, RTS="vrs")
dea.merge(x,y,M, RTS="crs")
```

---

dea.plot

*Plot of DEA technologies*

---

## Description

Draw a graph of a DEA technology. Designed for two goods illustrations, either isoquant (2 inputs), transformation curve (2 outputs), or a production function (1 input and 1 output). If the number of good is larger than 2 then aggregation occur, either simple or weighted.

## Usage

```
dea.plot(x, y, RTS="vrs", ORIENTATION="in-out", txt=NULL, add=FALSE,
        wx=NULL, wy=NULL, TRANSPOSE=FALSE, fex=1, GRID=FALSE,
        RANGE=FALSE, param=NULL, ..., xlim, ylim, xlab, ylab)

dea.plot.frontier(x, y, RTS="vrs",...)

dea.plot.isoquant(x1, x2, RTS="vrs",...)

dea.plot.transform(y1, y2, RTS="vrs",...)
```

## Arguments

**x** The good illustrated on the first axis. If there are more than 1 input then inputs are just summed or, if `wx` is present, a weighted sum of inputs is used.

**y** The good illustrated on the second axis. If there are more than 1 output then outputs are just summed or, if `wy` is present, a weighted sum of outputs is used.

x1, y1	The good illustrated on the first axis
x2, y2	The good illustrated on the second axis
RTS	Underlying DEA model / assumptions about returns to scale: "fdh" (0), "vrs" (1), "drs" (2), "crs" (3), "irs" (4), "irs2" (5) (irs without convexity), "add" (6), and "fdh+" (7). Numbers in parenthesis can also be used as values for RTS
ORIENTATION	Input-output graph of 1 input and 1 output is "in-out" (0), graph of 2 inputs is "in" (1), and graph of 2 outputs is "out" (2).
txt	txt is an array to label the observations. If txt=TRUE the observations are labeled by the observation number or rownames if there are any.
add	For add=T the technology is drawn on top of an existing graph. With the default add=F, a new graph is made.
wx	Weight to aggregate the first axis if there are more than 1 good behind the first axis.
wy	Weights to aggregate for the second axis if there are more than 1 good behind the second the second axis.
TRANSDPOSE	Only relevant for more than 1 good for each axis, see <a href="#">dea</a> for a description of this option.
GRID	If GRID=TRUE a gray grid is put on the plot.
...	Usual options for the methods plot, lines, and abline etc.
fex	Relative size of the text/labels on observations; corresponds to cex, but only changes the size of the text.
RANGE	A logical variable, if RANGE=TRUE the limits for the graph is the range of the variables; zero is always included. Default is RANGE=FALSE when the range is from zero to the max values. Relevant if some values are negative.
param	Possible parameters. At the moment only used for RTS="fdh+"; see the section details and examples for its use. Future versions might also use param for other purposes.
xlim	Possible limits c(x1, x2) for the first axis
ylim	Possible limits c(y1, y2) for the second axis
xlab	Possible label for the x-axis
ylab	Possible label for the y-axis

## Details

The method `dea.plot` is the general plotting method. The the 3 others are specialized versions for frontiers (1 input and 1 output), isoquant curves (2 inputs for given outputs), and transformation curves (2 outputs for given inputs) obtained by using the argument `ORIENTATION`.

The `crs` factor in `RTS="fdh+"` that sets the lower and upper bound can be changed by the argument `param` that will set the lower and upper bound to `1-param` and `1+param`; the default value is `param=.15`. The value must be greater than or equal to 0 and strictly less than 1. A value of 0 corresponds to `RTS="fdh"`. The FDH+ technology set is described in Bogetoft and Otto (2011) pages 72–73.

**Value**

No return, uses the original graphing system.

**Note**

If there are more than 1 good for the arguments `x` and `y` then the goods are just summed or, if `wx` or `wy` are present, weighted sum of goods are used. In this case the use of the command `identify` must be called as `dea.plot(rowSums(x), rowSums(y))`.

*Warning* If you use this facility to plot multi input and multi output then the plot may deceive you as fully multi efficient firms are not necessarily placed on the two dimensional frontier.

Note that `RTS="add"` and `RTS="fdh+"` only works for `ORIENTATION="in-out"` (0).

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**References**

Peter Bogetoft and Lars Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011

Paul Murrell; *R Graphics*; Chapman & Hall 2006

**See Also**

The documentation for the function `plot` and Murrell (2006) for further options and on customizing plots.

**Examples**

```
x <- matrix(c(100,200,300,500,600,100), ncol=1)
y <- matrix(c(75,100,300,400,400,50), ncol=1)

dea.plot(x,y,RTS="vrs",ORIENTATION="in-out",txt=LETTERS[1:length(x)])
dea.plot(x,y,RTS="crs",ORIENTATION="in-out",add=TRUE,lty="dashed")

dea.plot.frontier(x,y,txt=1:dim(x)[1])

n <- 10
x <- matrix(1:n,,1)
y <- matrix(x^(1.6) + abs(rnorm(n)),,1)
dea.plot.frontier(x,y,RTS="irs",txt=1:n)
dea.plot.frontier(x,y,RTS="irs2",add=TRUE,lty="dotted")

# Two different forms of irs: irs and irs2, and two different ways to
# make a frontier
id <- sample(1:n,30,replace=TRUE)
dea.plot(x[id],y[id],RTS="irs",ORIENTATION="in-out")
dea.plot.frontier(x[id],y[id],RTS="irs2")

# Difference between the FDH technology and the additive
```

```

# FRH technology
x <- matrix(c(100,220,300,520,600,100),ncol=1)
y <- matrix(c(75,100,300,400,400,50),ncol=1)
dea.plot(x,y,RTS="fdh",ORIENTATION="in-out",txt=LETTERS[1:length(x)])
dea.plot(x,y,RTS="add",ORIENTATION="in-out",add=TRUE,lty="dashed",lwd=2)
dea.plot(x,y,RTS="fdh+",ORIENTATION="in-out",add=TRUE,
         lty="dotted",lwd=3,col="red")

# Use of parameter in FDH+
dea.plot(x,y,RTS="fdh",ORIENTATION="in-out",txt=LETTERS[1:length(x)])
dea.plot(x,y,RTS="fdh+",ORIENTATION="in-out",add=TRUE,lty="dashed")
dea.plot(x,y,RTS="fdh+",ORIENTATION="in-out",add=TRUE,lty="dotted",param=.5)

```

---

eff, efficiencies	<i>Calculate efficiencies for Farrell and sfa object</i>
-------------------	--

---

## Description

Calculate efficiencies for Farrell and sfa object. For a sfa there are several types

## Usage

```

eff( object, ... )
efficiencies( object, ... )
## Default S3 method:
efficiencies( object, ... )
## S3 method for class 'Farrell'
efficiencies(object, type = "Farrell", ...)
## S3 method for class 'Farrell'
eff(object, type = "Farrell", ...)
## S3 method for class 'sfa'
efficiencies(object, type = "BC", ...)
## S3 method for class 'sfa'
eff(object, type = "BC", ...)

```

## Arguments

object	A Farrell object returned from a DEA function like <a href="#">dea</a> , <a href="#">sdea</a> , or <a href="#">mea</a> or an sfa object returned from the function <a href="#">sfa</a> .
type	The type of efficiencies to be calculated. For a Farrell object the possibilities are “Farrell” efficiency or “Shephard” efficiency. For a sfa object the possibilities are “BC”, “Mode”, “J”, or “add”.
...	Further arguments ...

## Details

The possible types for class `Farrell` (an object returned from `dea` et al. are “Farrell” and “Shephard”.

The possible types for class `sfa` efficiencies are

**BC** Efficiencies estimated by minimizing the mean square error; Eq. (7.21) in Bogetoft and Otto (2011, 219) and Battese and Coelli (1988, 392)

**Mode** Efficiencies estimates using the conditional mode approach; Bogetoft and Otto (2011, 219), Jondrow et al. (1982, 235).

**J** Efficiencies estimates using the conditional mean approach Jondrow et al. (1982, 235).

**add** Efficiency in the additive model, Bogetoft and Otto (2011, 219)

## Value

The efficiencies are returned as an array.

## Note

For the `Farrell` object the orientation is determined by the calculations that led to the object and can not be changed here.

## Author(s)

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

## References

Bogetoft and Otto; *Benchmarking with DEA, SFA, and R*, Springer 2011

## See Also

`dea` and `sfa`.

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.
```



---

eff.dens	<i>Estimate and plot density of efficiencies</i>
----------	--

---

### Description

A method to estimate and plot kernel estimate of (Farrell) efficiencies taken into consideration that efficiencies are bounded either above (input direction) or below (output direction).

### Usage

```
eff.dens(eff, bw = "nrd0")
```

```
eff.dens.plot(obj, bw = "nrd0", ..., xlim, ylim, xlab, ylab)
```

### Arguments

eff	Either a list of (Farrell) efficiencies or a Farrell object returned from the method <a href="#">dea</a> .
bw	Bandwidth, look at the documentation of <code>density</code> for an explanation.
obj	Either an array of efficiencies or a list returned from <code>eff.dens</code> .
...	Further arguments to the plot method like line type and line width.
xlim	Range on the x-axis; usually not needed, just use the defaults.
ylim	Range on the x-axis; usually not needed, just use the defaults.
xlab	Label for the x-axis.
ylab	Label for the y-axis.

### Details

The calculation is based on a reflection method (Silverman 1986, 30) using the default window kernel and default bandwidth (window width) in the method `density`.

The method `eff.dens.plot` plot the density directly, and `eff.dens` just estimate the numerical density, and the result can then either be plotted by `plot`, corresponds to `eff.dens.plot`, or by lines as an overlay on an existing plot.

### Value

The return from `eff.dens` is a list `list(x,y)` with efficiencies and the corresponding density values.

### Note

The input efficiency is also bounded below by 0, but for normal firms an efficiency at 0 will not happen, i.e. the boundary is not effective, and therefore this boundary is not taken into consideration.

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**References**

B.W. Silverman (1986), *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London.

**Examples**

```
e <- 1 - rnorm(100)
e[e>1] <- 1
e <- e[e>0]
eff.dens.plot(e)

hist(e, breaks=15, freq=FALSE, xlab="Efficiency", main="")
den <- eff.dens(e)
lines(den, lw=2)
```

---

eladder

*Efficiency ladder for a single firm*

---

**Description**

How the efficiency changes as the most influential peer is removed sequentially one at a time

**Usage**

```
eladder(n, X, Y, RTS = "vrs", ORIENTATION = "in",
        XREF=NULL, YREF=NULL, DIRECT = NULL, param=NULL, MAXELAD=NULL)
eladder.plot(elad, peer, TRIM = NULL, ...)
```

**Arguments**

n	The number of the firm where the ladder is calculated
X	Inputs of firms to be evaluated, a K x m matrix of observations of K firms with m inputs (firm x input). In case TRANSPOSE=TRUE the input matrix is transposed to input x firm.
Y	Outputs of firms to be evaluated, a K x n matrix of observations of K firms with n outputs (firm x input). In case TRANSPOSE=TRUE the output matrix is transposed to output x firm.
RTS	Text string or a number defining the underlying DEA technology / returns to scale assumption, see the possible values for <a href="#">dea</a> .
ORIENTATION	Input efficiency "in" (1), output efficiency "out" (2), and graph efficiency "graph" (3). For use with DIRECT, an additional option is "in-out" (0).
XREF	Inputs of the firms determining the technology, defaults to X

YREF	Outputs of the firms determining the technology, defaults to Y
DIRECT	Directional efficiency, DIRECT is either a scalar, an array, or a matrix with non-negative elements. See <a href="#">dea</a> for a further description of this argument.
param	Possible parameters. At the moment only used for RTS="fdh+" to set low and high values for restrictions on lambda; see the section details and examples in <a href="#">dea</a> for its use. Future versions might also use param for other purposes.
MAXELAD	The maximum number of influential peers to remove.
e1ad	The sequence of efficiencies returned from eladder.
peer	The sequence of peers returned from eladder.
TRIM	The number of characters for the name of the peers on the axis in the plot.
...	Usual options for the method plot.

### Details

The function `e1adder` calculates how the efficiency for a firm changes when the most influential peer is removed sequentially one at a time. Somewhere in the sequence the firm becomes efficient and are itself removed from the set of firms generating the technology (or the only firm left) and thereafter the efficiencies are super-efficiencies.

### Value

The object returned from `e1adder` is a list with components

eff	The sequence of efficiencies when the peer with the largest value of lambda has been removed.
peer	The sequence of removed peers corresponding to the largest values of lambda as index in the X rows.

### Note

When the number of firms is large then the number of influential peers will also be large and the names or numbers of the peers on the x-axis might be squeeze together and be illegible. In this case restrict the number of influential peers to be removed.

The efficiency step ladder is discussed in Essay 4 of Dag Fjeld Edvardsens's Ph.D. thesis from 2004.

### Author(s)

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

### References

Dag Fjeld Edvardsen; *Four Essays on the Measurement of Productive Efficiency*; University of Gothenburg 2004; <http://hdl.handle.net/2077/2923>

**Examples**

```

data(charnes1981)
x <- with(charnes1981, cbind(x1,x2,x3,x4,x5))
y <- with(charnes1981, cbind(y1,y2,y3))

# Choose the firm for analysis, we choose 'Tacoma'
n <- which(charnes1981$name=="Tacoma")[1]

el <- eladder(n, x, y, RTS="crs")
eladder.plot(el$eff, el$peer)

# Restrict to 20 most influential peers for 'Tacoma' and use names
# instead of number
eladder.plot(el$eff[1:20], charnes1981$name[el$peer][1:20])

# Truncate the names of the peers and put a title on top
eladder.plot(el$eff[1:20], charnes1981$name[el$peer][1:20], TRIM=5)
title("Eladder for Tacoma")

```

---

excess

*Excess input compared over frontier input*


---

**Description**

Excess input compared over frontier input and/or less output than frontier/transformation/optimal output.

**Usage**

```
excess(object, X = NULL, Y = NULL)
```

**Arguments**

object	A Farrell object as returned from functions like <a href="#">dea</a> , <a href="#">dea.direct</a> , <a href="#">linksdea</a> , and <a href="#">mea</a> .
X	Input matrix, only necessary for ordinary input Farrell efficiency
Y	Output matrix, only necessary for ordinary output Farrell efficiency

**Details**

For Farrell input efficiency  $E$  the excess input is  $(1 - E)X$  and for Farrell output efficiency  $F$  the missing output is  $(F - 1)Y$ .

For directional efficiency  $e$  in the direction  $D$  the excess input is  $eD$ .

If a firm is outside the technology set, as could be the case when calculating super-efficiencies, the Farrell input efficiency is larger than 1 and then the excess values are negative.

**Value**

Return a matrix with excess input and/or less output.

**Author(s)**

Peter Bogeroft and Lars Otto <larsot23@gmail.com>

**References**

Peter Bogetoft and Lars Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011

**Examples**

```
x <- matrix(c(100,200,300,500,100,200,600),ncol=1)
y <- matrix(c(75,100,300,400,25,50,400),ncol=1)

e <- dea(x,y)
excess(e,x)
x - eff(e) * x

e <- dea(x,y, ORIENTATION="graph")
excess(e, x, y)
x - eff(e) * x
1/eff(e) * y -y

me <- mea(x,y)
excess(me)
```

---

lambda

*Lambdas or the weight of the peers*

---

**Description**

The lambdas, i.e. the weight of the peers, for each firm.

**Usage**

```
lambda(object, KEEPREF = FALSE)
lambda.print(x, KEEPREF = FALSE, ...)
```

**Arguments**

object,x	A Farrell object as returned from <code>dea</code> et al.
KEEPREF	if TRUE then all firms are kept as reference firms even though they have all zero weights (lambda); might come handy if one needs to calculate $X \times \text{lambda}$ such that the firms in $X$ and $\text{lambda}$ agree. If FALSE, the default, then only weight for the peers are in the matrix $\text{lambda}$ .
...	Optional parameters for the print method.

**Details**

Only returns the the lambdas for firms that appear as a peer, i.e. only lambdas for firms where at least one element of the lambda is positive.

**Value**

The return is a matrix with the firms as rows and the peers as columns.

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**See Also**

[dea](#)

**Examples**

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.
```

---

make.merge

*Make an aggregation matrix to perform mergers*

---

**Description**

Make an aggregation matrix to perform mergers of firms. The matrix can be post multiplied (matrix multiplication) to input and output matrices to make merged input and output matrices.

**Usage**

```
make.merge(grp, nFirm = NULL, X = NULL, names = NULL)
```

**Arguments**

grp	Either a list of length $K_g$ for $K_g$ firms after mergers; each component of the list is a (named) list with the firm numbers or names going into this merger. Or a factor of length $K$ with $K_g$ levels where where each level determines a merger; to exclude firms for mergers set the factor value to NA.
nFirm	Number of firms before the mergers
X	A matrix of inputs or outputs where the rows corresponds to the number of original (starting) firms
names	A list with names of all firms, only needed if the mergers are given as a list of names, i.e. grp is a list of names.

**Details**

Either nFirm or X must be present; if both are present then nFirm must be equal to the number of rows in X, the number of firms.

When X is an input matrix of dimension  $K \times m$ , K firms and m inputs, and `M <- make.merge(gr,K)` then `M %*% X` is the input matrix for the merged firms.

**Value**

Returns an aggregation matrix of dimension Kg times K where rows corresponds to new merged firms and columns are 1 for firms to be included and 0 for firms to be excluded in the the given merger as defined by the row.

**Note**

The argument TRANSPOSE has not been implemented for this function. If you need transposed matrices you must transpose the merger matrix yourself. If you define mergers via factors there is no need to transpose in the arguments; just do not use X in the arguments.

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**See Also**

[dea.merge](#)

**Examples**

```
# To merge firms 1,3, and 5; and 2 and 4 of 7 firms into 2 new firms
# the aggregation matrix is M; not all firms are involved in a merger.
M <- make.merge(list(c(1,3,5), c(2,4)),7)
print(M)

# Merge 1 and 2, and 4 and 5, and leave 3 alone, total of 5 firms.
# Using a list
M1 <- make.merge(list(c(1,2), c(4,5)), nFirm=5)
print(M1)

# Using a factor
fgr <- factor(c("en","en",NA,"to","to"))
M2 <- make.merge(fgr)
print(M2)

# Name of mergers
M3 <- make.merge(list(AB=c("A","B"), DE=c("D","E")), names=LETTERS[1:5])
print(M3)

# No name of mergers
M4 <- make.merge(list(c("A","B"), c("D","E")), names=LETTERS[1:5])
print(M4)
```

---

 malmq

*Malmquist index*


---

### Description

Estimates Malmquist indices for productivity and its decomposition between two periods. The units in the two periods does not have to be exactly the same, but the Malmquist index is only calculated for units present in both periods.

### Usage

```
malmq(X0, Y0, ID0 = NULL, X1, Y1, ID1 = NULL, RTS = "vrs", ORIENTATION = "in",
      SLACK = FALSE, DUAL = FALSE, DIRECT = NULL, param = NULL, TRANSPOSE = FALSE,
      FAST = TRUE, LP = FALSE, CONTROL = NULL, LPK = NULL)
```

### Arguments

X0	Inputs of firms in period 0, a K0 x m matrix of observations of K0 firms with m inputs (firm x input).
Y0	Outputs of firms in period 0, a K0 x n matrix of observations of K0 firms with n outputs (firm x input).
ID0	Index for firms in period 0; could be numbers or labels. Length K0.
X1	Inputs of firms in period 1, a K1 x m matrix of observations of K1 firms with m inputs (firm x input).
Y1	Outputs of firms in period 1, a K1 x n matrix of observations of K1 firms with n outputs (firm x input).
ID1	Index for firms in period 0; could be numbers or labels. Length K0.
RTS	Returns to scale assumption as in <a href="#">dea</a> .
ORIENTATION	Input efficiency "in" (1), output efficiency "out" (2), and graph efficiency "graph" (3) as in <a href="#">dea</a> .
SLACK	See <a href="#">dea</a> .
DUAL	See <a href="#">dea</a> .
DIRECT	See <a href="#">dea</a> .
param	See <a href="#">dea</a> .
TRANSPOSE	See <a href="#">dea</a> .
FAST	See <a href="#">dea</a> .
LP	See <a href="#">dea</a> .
CONTROL	See <a href="#">dea</a> .
LPK	See <a href="#">dea</a> .

### Details

See ... \ for details



**Value**

m	Malmquist index for productivity.
tc	Index for technology change.
ec	Index for efficiency change.
mq	Malmquist index for productivity; same as m.
id	Index for firms present in both period 0 and period 1.
id0	Index for firms in period 0 that are also in period 1.
id1	Index for firms in period 1 that are also in period 0.
e00	The efficiencies for period 0 with reference technology from period 0.
e10	The efficiencies for period 1 with reference technology from period 0.
e11	The efficiencies for period 1 with reference technology from period 1.
e01	The efficiencies for period 0 with reference technology from period 1.

**Note**

The calculations of efficiencies are only done for units present in both periods.

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**References**

Peter Bogetoft and Lars Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011

**See Also**

[dea](#)

**Examples**

```
x0 <- matrix(c(10, 28, 30, 60),ncol=1)
y0 <- matrix(c(5, 7, 10, 15),ncol=1)
x1 <- matrix(c(12, 26, 16, 60 ),ncol=1)
y1 <- matrix(c(6, 8, 9, 15 ),ncol=1)

dea.plot(x0, y0, RTS="vrs", txt=TRUE)
dea.plot(x1, y1, RTS="vrs", add=TRUE, col="red")
points(x1, y1, col="red", pch=16)
text(x1, y1, 1:dim(x1)[1], col="red", adj=-1)

m <- malmq(x0,y0,,x1,y1,,RTS="vrs")
print("Malmquist index for change in productivity, technology change:")
print(m$mq)
print("Index for change of frontier:")
print(m$tc)
```

---

malmquist

*Malmquist index for firms in a panel*


---

### Description

Estimate Malmquist index for firms in a panel data set. The data set does not need to be balanced.

### Usage

```
malmquist(X, Y, ID, TIME, RTS = "vrs", ORIENTATION = "in", SLACK = FALSE, DUAL = FALSE,
DIRECT = NULL, param = NULL, TRANSPOSE = FALSE, FAST = TRUE, LP = FALSE,
CONTROL = NULL, LPK = NULL)
```

### Arguments

X	Inputs of firms in many periods, a (T*K) x m matrix of observations of K firms with m outputs (firm x input) in at the most T periods.
Y	Outputs of firms in many periods, a (T*K) x n matrix of observations of K0 firms with n outputs (firm x input) in at the most T periods.
ID	Identifier for the firms in rows of X and Y.
TIME	Array with period number for each row in the input maxtrix X and output matrixY
RTS	Returns to scale assumption as in <a href="#">dea</a> .
ORIENTATION	Input efficiency "in" (1), output efficiency "out" (2), and graph efficiency "graph" (3) as in <a href="#">dea</a> .
SLACK	See <a href="#">dea</a> .
DUAL	See <a href="#">dea</a> .
DIRECT	See <a href="#">dea</a> .
param	See <a href="#">dea</a> .
TRANSPOSE	See <a href="#">dea</a> .
FAST	See <a href="#">dea</a> .
LP	See <a href="#">dea</a> .
CONTROL	See <a href="#">dea</a> .
LPK	See <a href="#">dea</a> .

### Details

Malmquist uses [ma1mq](#) for the calculations of the necessary efficiencies.

**Value**

m	Malmquist indicies, an array of length T*K in the order of ID and TIME, i.e. the order of the rows of X.
tc	Technical change indices, an array of length T*K.
ec	Efficiency indices, an array of length T*K.
id	Index for firms as ID
time	Index for time as TIME
e00	The efficiencies for period 0 with reference technology from period 0.
e10	The efficiencies for period 1 with reference technology from period 0.
e11	The efficiencies for period 1 with reference technology from period 1.
e01	The efficiencies for period 0 with reference technology from period 1.

**Note**

The lagged values e11 are not necessary equal to values of e00 as the reference technology for the two periods could be generated by different units, if the units in different time periodes are not the same.

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**References**

Peter Bogetoft and Lars Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011

**See Also**

[dea](#), [malmq](#)

**Examples**

```
x0 <- matrix(c(10, 28, 30, 60),ncol=1)
y0 <- matrix(c(5, 7, 10, 15),ncol=1)
x1 <- matrix(c(12, 26, 16, 60 ),ncol=1)
y1 <- matrix(c(6, 8, 9, 15 ),ncol=1)

dea.plot(x0, y0, RTS="vrs", txt=TRUE)
dea.plot(x1, y1, RTS="vrs", add=TRUE, col="red")
points(x1, y1, col="red", pch=16)
text(x1, y1, 1:dim(x1)[1], col="red", adj=-1)

X <- rbind(x0, x1)
Y <- rbind(y0, y1)
ID <- rep(1:dim(x1)[1], 2)
TIME <- c(rep(0,dim(x1)[1]), rep(1,dim(x1)[1]))
mq <- malmquist(X,Y, ID, TIME=TIME)
mq$tc # Technical change
mq$m # Malmquist index for productivity change
```

mea

*MEA multi-directional efficiency analysis***Description**

Potential improvements PI or multi-directional efficiency analysis. The result is an excess value measures by the direction.

The direction is determined by the direction corresponding to the minimum input/maximum direction each good can be changed when they are changed one at a time.

**Usage**

```
mea(X, Y, RTS = "vrs", ORIENTATION = "in", XREF = NULL, YREF = NULL,
    FRONT.IDX = NULL, param=NULL, TRANSPOSE = FALSE,
    LP = FALSE, CONTROL = NULL, LPK = NULL)
mea.lines(N, X, Y, ORIENTATION="in")
```

**Arguments**

X	K times m matrix with K firms and m inputs as in dea
Y	K times n matrix with K firms and n outputs as in dea
RTS	Text string or a number defining the underlying DEA technology / returns to scale assumption.
0	fdh Free disposability hull, no convexity assumption
1	vrs Variable returns to scale, convexity and free disposability
2	drs Decreasing returns to scale, convexity, down-scaling and free disposability
3	crs Constant returns to scale, convexity and free disposability
4	irs Increasing returns to scale, (up-scaling, but not down-scaling), convexity and free disposability
6	add Additivity (scaling up and down, but only with integers), and free disposability
7	fdh+ A combination of free disposability and restricted or local constant return to scale
ORIENTATION	Input efficiency "in" (1) or output efficiency "out" (2), and also the additional option "in-out" (0) for both input and output direction.
XREF	Inputs of the firms determining the technology, defaults to X
YREF	Outputs of the firms determining the technology, defaults to Y
FRONT.IDX	Index for firms determining the technology
param	Possible parameters. At the moment only used for RTS="fdh+" to set low and high values for restrictions on lambda; see the section details and examples in <a href="#">dea</a> for its use. Future versions might also use param for other purposes.
TRANSPOSE	as in dea
LP	as in dea
CONTROL	as in dea

LPK	as in dea
N	Number of firms where directional lines are to be drawn on an already existing frontier plot ( <a href="#">dea.plot.frontier</a> )

### Details

Details can be found in Bogetoft and Otto (2011, 121–124).

This method is for input directional efficiency only interesting when there are 2 or more inputs, and for output only when there are 2 or more outputs.

### Value

The results are returned in a Farrell object with the following components.

eff	Excess value in DIRECT units of measurement, this is <i>not</i> Farrell efficiency
lambda	The lambdas, i.e. the weight of the peers, for each firm
objval	The objective value as returned from the LP program; normally the same as eff
RTS	The return to scale assumption as in the option RTS in the call
ORIENTATION	The efficiency orientation as in the call
direct	A $K$ times $m \times (n + n)$ matrix with directions for each firm: the number of columns depends on whether it is input, output or in-out orientated.
TRANSDPOSE	As in the call

### Note

The calculation is done in [dea](#) after a calculation of the direction that then is used in the argument DIRECT. The calculation of the direction is done in a series LP programs, one for each good in the direction.

### Author(s)

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

### References

Peter Bogetoft and Lars Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011

### See Also

[dea](#) and the argument DIRECT.

**Examples**

```

X <- matrix(c(2, 2, 5, 10, 10, 3, 12, 8, 5, 4, 6,12), ncol=2)
Y <- matrix(rep(1,dim(X)[1]), ncol=1)

dea.plot.isoquant(X[,1], X[,2],txt=1:dim(X)[1])
mea.lines(c(5,6),X,Y)

me <- mea(X,Y)
me
peers(me)
# MEA potential saving in inputs, exces inputs
eff(me) * me$direct
me$eff * me$direct

# Compare to traditionally Farrell efficiency
e <- dea(X,Y)
e
peers(e)
# Farrell potential saving in inputs, exces inputs
(1-eff(e)) * X

```

---

milkProd

*Data: Milk producers*


---

**Description**

Data collected from Danish milk producers.

**Usage**

```
data(milkProd)
```

**Format**

A data frame with 108 observations on the following 5 variables.

```

farmNo farm number
milk Output of milk, kg
energy Energy expenses
vet Veterinary expenses
cows Number of cows

```

**Note**

Data as .csv are loaded by the command `data` using `read.table(..., header = TRUE, sep = ";")` such that this file is a semicolon separated file and not a comma separated file.

**Source**

Accounting and buissness check data

**Examples**

```
data(milkProd)
y <- with(milkProd, cbind(milk))
x <- with(milkProd, cbind(energy, vet, cows))
```

---

norWood2004

*Data: Forestry in Norway*

---

**Description**

A data set for 113 farmers in forestry in Norway.

**Usage**

```
data(norWood2004)
```

**Format**

A data frame with 113 observations on the following 7 variables.

firm firm number  
m Variable cost  
x Woodland, value of forrest and land  
y Profit  
z1 Secondary income from ordinary farming  
z3 Age of forrest owner  
z6 Whether there is a long-term plan =1 or not =0

**Details**

Collected from farmers in forestry.

**Note**

Data as .csv are loaded by the command `data` using `read.table(..., header=TRUE, sep=";")` such that this file is a semicolon separated file and not a comma separated file.

**Source**

Norwegian Agricultural Economics Research Institute.

**Examples**

```
data(norWood2004)
## maybe str(norWood2004) ; plot(norWood2004) ...
```

---

outlier.ap

*Detection of outliers in benchmark models*


---

### Description

The functions implements the Wilson (1993) outlier detection method using only R functions.

### Usage

```
outlier.ap(X, Y, NDEL = 3, NLEN = 25, TRANSPPOSE = FALSE)
```

```
outlier.ap.plot(ratio, NLEN = 25, xlab = "r", ylab = "Log ratio",
               ..., ylim)
```

### Arguments

X	Input as a firms times goods matrix, see TRANSPPOSE.
Y	Output as a firms times goods matrix, see TRANSPPOSE.
NDEL	The maximum number of firms to be considered as a group of outliers, i.e. the maximum number of firms to be deleted.
NLEN	The number of ratios to save for each level or removal, the number of rows in ratio used.
TRANSPPOSE	Input and output matrices are treated as firms times goods matrices for the default value TRANSPPOSE=FALSE corresponding to the standard in R for statistical models. When TRUE data matrices are transposed to good times firms matrices as is normally used in LP formulation of the problem.
ratio	The ratio component from the list as output from outlier.ap.
xlab	Label for the x-axis.
ylab	Label for the y-axis
ylim	The y limits (y1, y2) of the plot, an array/vector of length 2.
...	Usual options for the methods plot and lines.

### Details

An implementation of the method in Wilson (1993) using only R functions and especially the function det to calculate  $R_{\min}^{(i)}$ .

An elementary presentation of the method is found in Bogetoft and Otto (2011), Sect. 5.13 on outliers.

### Value

ratio	A $\min(\text{NLEN}, K) \times \text{NDEL}$ matrix with the log-ratios to be plotted.
imat	A $\text{NDEL} \times \text{NDEL}$ matrix with indices for deleted firms.
r0	A $\text{NDEL}$ array with the minimum value $R^i$ of the for each number of deleted firms.



**Note**

The function `outlier.ap` is extremely slow and for NDEL larger than 3 or 4 it might be advisable to use the function `ap` from the package **FEAR**.

The name of the returned components are the same as for `ap` in the package **FEAR**.

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**References**

Bogetoft and Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011

Wilson (1993), "Detecting outliers in deterministic nonparametric frontier models with multiple outputs," *Journal of Business and Economic Statistics* 11, 319-323.

Wilson (2008), "FEAR 1.0: A Software Package for Frontier Efficiency Analysis with R," *Socio-Economic Planning Sciences* 42, 247-254

**See Also**

The function `ap` in the package **FEAR**.

**Examples**

```
n <- 25
x <- matrix(rnorm(n))
y <- .5 + 2.5*x + 2*rnorm(25)
tap <- outlier.ap(x,y, NDEL=2)
print(cbind(tap$imat,tap$rmin), na.print="", digit=2)
outlier.ap.plot(tap$ratio)
```

---

peers

*Find peer firms and units*

---

**Description**

The function `peers` finds for each firm its peers, `get.number.peers` finds for each peer the number of times this peer appears as a peer, and `get.which.peers` determines for one or more peers the firms they appear as peers for.

**Usage**

```
peers(object, NAMES = FALSE)
get.number.peers(object, NAMES = FALSE)
get.which.peers(object, N = 1:dim(object$lambda)[2])
```

**Arguments**

object	An object of class Farrell as returned by the functions <a href="#">dea</a> , <a href="#">dea.direct</a> et al.
NAMES	If true then names for the peers are returned if names are available otherwise the unit index numbers are used. If NAMES is a list of names with length equal to the number of units then it is used as names for peers.
N	The peer(s) for which

**Details**

The returned values are index of the firms and can be used by itself, but can also be used as an index for a variable with names of the firms.

The `peers` returns a matrix with numbers for the peers for each firm; for firms with efficiency 1 the peers are just the firm itself. If there is slack in the evaluation of a firm with efficiency 1, this can be found with a call to [slack](#), either directly or by the argument SLACK when a function [dea](#) was called to generate the Farrell object.

The `get.number.peers` returns the number of firms that a peer serves as a peer for.

**Value**

The return values are firm numbers. If the argument NAMES=TRUE is used in the function `peers` the return is a list of names of the peers if names for the firms are available as rownames.

**Note**

Peers are defined as firms where the corresponding lambdas are positive.

Note that peers might change between a Farrell object return from `dea` with SLACK=FALSE and a call with SLACK=TRUE or a following call to the function `slack` because a peer on the frontier with slack might by the call to `dea` be seen as a peer for itself whereas this will not happen when slacks are calculated.

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**References**

Peter Bogetoft and Lars Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011. Sect. 4.6 page 93

**See Also**

[dea](#)

**Examples**

```
x <- matrix(c(100,200,300,500,100,200,600),ncol=1)
y <- matrix(c(75,100,300,400,25,50,400),ncol=1)

e <- dea(x,y)
peers(e)
get.number.peers(e)

# Who are the firms that firm 1 and 4 is peers for
get.which.peers(e, c(1,4))
```

---

pigdata

*Data: Multi-output pig producers*

---

**Description**

Input and output data for 248 pig producers that also produces crop, i.e. a multi-output data set.

**Usage**

```
data(pigdata)
```

**Format**

A data frame with 248 observations on the following 16 variables.

```
firm Serial number for pig producer
x1 Input fertilizer
x2 Input feedstuf
x3 Input land
x4 Input labour
x5 Input machinery
x6 Input other capital
y2 Output crop
y4 Output pig
w1 Price of fertilizer
w2 Price of feedstuf
w3 Price of land
w4 Price of labour
w5 Price of michenery
w6 Price of other capital
p2 Price of pig
p4 Price of crop
cost Total cost,  $w1*x1+...+w6*x6$ .
rev Total revenue,  $p2*y2+p4*y4$ .
```

**Details**

In raising pigs, most farmers also produce crops to feed the pigs. Labor and capital are used not just directly for pig-raising but also on the field.

**Note**

Data as .csv are loaded by the command `data` using `read.table(..., header = TRUE, sep = ";")` such that this file is a semicolon separated file and not a comma separated file.

**Source**

Farmers accounting data converted to index.

**Examples**

```
data(pigdata)
## maybe str(pigdata) ; plot(pigdata) ...
```

---

projekt	<i>Data: Milk producers</i>
---------	-----------------------------

---

**Description**

Accounting and production data for 101 milk producing farmers.

**Usage**

```
data(projekt)
```

**Format**

A data frame with 101 observations on the following 14 variables.

`numb` Serial number for the milk producer

`cows` Number of cows

`vet` Veterinary expences

`unitCost` Unit cost, variable cost

`capCost` Capacity cost

`fixedCost` Fixed cost

`milkPerCow` Milk per cow, kg

`quota` Milk quota

`fatPct` Fat percent in milk

`protPct` Protein percent in milk

`cellCount` Cell count for milk

`race` Race for cows, a factor with levels jersey, large, and mixed

`type` Type of production, conventional or organic, a factor with levels conv orga

`age` Age of the farmer

**Details**

Data is a mix of accounting data and production controls.

**Note**

Data as .csv are loaded by the command `data` using `read.table(..., header = TRUE, sep = ";")` such that this file is a semicolon separated file and not a comma separated file.

**Source**

Collected from farmers.

**Examples**

```
data(projekt)
## maybe str(projekt) ; plot(projekt) ...
```

---

sdea	<i>Super efficiency</i>
------	-------------------------

---

**Description**

The method `sdea` calculates super-efficiency and returns the same class of object as `dea`.

**Usage**

```
sdea(X, Y, RTS = "vrs", ORIENTATION = "in", DIRECT = NULL, param = NULL,
      TRANSPOSE = FALSE, LP = FALSE)
```

**Arguments**

X	Inputs of firms to be evaluated, a $K \times m$ matrix of observations of $K$ firms with $m$ inputs (firm $\times$ input). In case <code>TRANSPOSE=TRUE</code> the input matrix is transposed to input $\times$ firm.
Y	Outputs of firms to be evaluated, a $K \times n$ matrix of observations of $K$ firms with $n$ outputs (firm $\times$ input). In case <code>TRANSPOSE=TRUE</code> the output matrix is transposed to output $\times$ firm.
RTS	Text string or a number defining the underlying DEA technology / returns to scale assumption; the same values as for <code>dea</code> .
0	fdh Free disposability hull, no convexity assumption
1	vrs Variable returns to scale, convexity and free disposability
2	drs Decreasing returns to scale, convexity, down-scaling and free disposability
3	crs Constant returns to scale, convexity and free disposability
4	irs Increasing returns to scale, (up-scaling, but not down-scaling), convexity and free disposability
5	irs2 Increasing returns to scale (up-scaling, but not down-scaling), additivity, and free disposability
6	add Additivity (scaling up and down, but only with integers), and free disposability
7	fdh+ A combination of free disposability and restricted or local constant return to scale

ORIENTATION	Input efficiency "in" (1), output efficiency "out" (2), and graph efficiency "graph" (3). For use with DIRECT, an additional option is "in-out" (0).
DIRECT	Directional efficiency, DIRECT is either a scalar, an array, or a matrix with non-negative elements. If the argument is a scalar, the direction is (1,1,...,1) times the scalar; the value of the efficiency depends on the scalar as well as on the unit of measurements. If the argument an array, this is used for the direction for every firm; the length of the array must correspond to the number of inputs and/or outputs depending on the ORIENTATION. If the argument is a matrix then different directions are used for each firm. The dimensions depends on the ORIENTATION, the number of firms must correspond to the number of firms in X and Y. DIRECT must not be used in connection with DIRECTION="graph".
param	Argument is at present only used when RTS="fdh+", see <a href="#">dea</a> for a description.
TRANSPOSE	See the description in <a href="#">dea</a> .
LP	Only for debugging, see the description in <a href="#">dea</a> .

### Details

Super-efficiency measures are constructed by avoiding that the evaluated firm can help span the technology; ie. if the firm in `qestuen` is a firm on the frontier in a normal `dea` approach then this firm in super efficiency might be outside the technology set.

### Value

The object returned is a Farrell object with the component described in [dea](#). The relevant components are

<code>eff</code>	The efficiencies. Note when DIRECT is used then the efficiencies are not Farrell efficiencies but rather excess values in DIRECT units of measurement
<code>lambda</code>	The lambdas, i.e. the weight of the peers, for each firm
<code>objval</code>	The objective value as returned from the LP program; normally the same as <code>eff</code> .
<code>RTS</code>	The return to scale assumption as in the option RTS in the call
<code>ORIENTATION</code>	The efficiency orientation as in the call

### Note

Calculation of slacks for super efficiency should be done by using the option `SLACK=TRUE` in the call of the method `sdea`. If the two phases are done in two steps as first a call to `sdea` and then a call to `slacks` the user must make sure to set the reference technology to the one corresponding to super-efficiency in the call to `slack` and this requires a loop with calls to `slack`.

### Author(s)

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

## References

Peter Bogetoft and Lars Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011. Sect. 5.2 page 115

P Andersen and NC Petersen; “A procedure for ranking efficient units in data envelopment analysis”; *Management Science* 1993 39(10):1261–1264

## See Also

[dea](#)

## Examples

```
x <- matrix(c(100,200,300,500,100,200,600),ncol=1)
y <- matrix(c(75,100,300,400,25,50,400),ncol=1)
se <- sdea(x,y)
se

# Leave out firm 3 as a determining firm of the technology set
n <- 3
dea.plot.frontier(x[-n], y[-n], txt=(1:dim(x)[1])[-n])
# Plot and label firm 3
points(x[n],y[n],cex=1.25,pch=16)
text(x[n],y[n],n,adj=c(-.75,.75))
```

---

sfa

*Stochastic frontier estimation*

---

## Description

Estimate a stochastic frontier production function using a maximum likelihood method.

## Usage

```
sfa(x, y, beta0 = NULL, lambda0 = 1, resfun = ebeta,
    TRANSPOSE = FALSE, DEBUG=FALSE,
    control=list(), hessian=2)
```

```
te.sfa(object)
teBC.sfa(object)
teMode.sfa(object)
teJ.sfa(object)

te.add.sfa(object)

sigma2u.sfa(object)
```

```
sigma2v.sfa(object)
sigma2.sfa(object)
```

```
lambda.sfa(object)
```

### Arguments

x	Input as a K x m matrix of observations on m inputs from K firms; (firm x input); MUST be a matrix. No constant for the intercept should be included in x as it is added by default.
y	Output; K times 1 matrix (one output)
beta0	Optional initial parameter values
lambda0	Optional initial ratio of variances
resfun	Function to calculate the residuals, default is a linear model with an intercept. Must be called as resfun(x, y, parm) where parm=c(beta, lambda) or parm=c(beta), and return the residuals as an array of length corresponding to the length of output y.
TRANSDPOSE	If TRUE, data is transposed, i.e. input is now m x K matrix
DEBUG	Set to TRUE to get various debugging information written on the console
control	List of control parameters to ucminf
hessian	How the Hessian is delivered, see the ucminf documentation
object	Object of class 'sfa' as output from the function sfa

### Details

The optimization is done by the R method `ucminf` from the package with the same name. The efficiency terms are assumed to be half-normal distributed.

Changing the maximum steplength, the trust region, might be important, and this can be done by the option `'control= list(stepmax=0.1)'`. The default value is 0.1 and that value is suitable for parameters around 1; for smaller parameters a lower value should be used. Notice that the step length is updated by the optimizing program and thus must be set for every call of the function `sfa` if it is to be set.

The generic functions `print.sfa`, `summary.sfa`, `fitted.sfa`, `residuals.sfa`, `logLik.sfa`, and `coef.sfa` all work as expected.

The methods `te.sfa`, `teMode.sfa` etc. calculates the efficiency corresponding to different methods

### Value

The values returned from `sfa` is the same as for `ucminf`, i.e. a list with components plus some especially relevant for `sfa`:

par	The best set of parameters found <code>c(beta, lambda)</code> .
value	The value of minus log-likelihood function corresponding to 'par'.
beta	The parameters for the function
sigma2	The estimate of the total variance



lambda	The estimate of lambda
N	The number of observations
df	The degrees of freedom for the model
residuals	The residuals as a K times 1 matrix/vector, can also be obtained by <code>residuals(sfa-object)</code>
fitted.values	Fitted values
vcov	The variance-covarians matrix for all estimated parameters incl. lambda
convergence	An integer code. '0' indicates successful convergence. Some of the error codes taken from <code>ucminf</code> are '1' Stopped by small gradient ( <code>grtol</code> ). '2' Stopped by small step ( <code>xtol</code> ). '3' Stopped by function evaluation limit ( <code>maxeval</code> ). '4' Stopped by zero step from line search More codes are found in <code>ucminf</code>
message	A character string giving any additional information returned by the optimizer, or 'NULL'.
o	The object returned by <code>ucminf</code> , for further information on this see <code>ucminf</code>

### Note

Calculation of technical efficiencies for each unit can be done by the method `te.sfa` as shown in the examples.

`te.sfa(sfaObject)`, `teBC.sfa(sfaObject)`: Efficiencies estimated by minimizing the mean square error; Eq. (7.21) in Bogetoft and Otto (2011, 219) and Battese and Coelli (1988, 392)

`teMode.sfa(sfaObject)`, `te1.sfa(sfaObject)`: Efficiencies estimates using the conditional mode approach; Bogetoft and Otto (2011, 219), Jondrow et al. (1982, 235).

`teJ.sfa(sfaObject)`, `te2.sfa(sfaObject)`: Efficiencies estimates using the conditional mean approach Jondrow et al. (1982, 235).

`te.add.sfa(sfaObject)` Efficiency in the additive model, Bogetoft and Otto (2011, 219)

The variance of the distribution of efficiency can be calculated by `sigma2u.sfa(sfaObject)`, the variance of the random error by `sigma2v.sfa(sfaObject)`, and the total variance (sum of variances of efficiency and random noise) by `sigma2.sfa`.

The ratio of variances of the efficiency and the random noise can be found from the method `lambda.sfa`

The generic method summary prints the parameters, standard errors, t-values, and a few more statistics from the optimization.

### Author(s)

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

### References

Bogetoft and Otto; *Benchmarking with DEA, SFA, and R*, Springer 2011; chapters 7 and 8.

**See Also**

See the method `ucminf` for the possible optimization methods and further options to use in the option control.

The method `sfa` in the package **frontier** gives another way to estimate stochastic production functions.

**Examples**

```
# Example from the book by Coelli et al.
# d <- read.csv("c:/work/rpack/front41Data.csv", header = TRUE, sep = ",")
# x <- cbind(log(d$capital), log(d$labour))
# y <- matrix(log(d$output))

n <- 50
x1 <- 1:50 + rnorm(50,0,10)
x2 <- 100 + rnorm(50,0,10)
x <- cbind(x1,x2)
y <- 0.5 + 1.5*x1 + 2*x2 + rnorm(n,0,1) - pmax(0,rnorm(n,0,1))
sfa(x,y)
summary(sfa(x,y))

# Estimate efficiency for each unit
o <- sfa(x,y)
eff(o)

te <- te.sfa(o)
teM <- teMode.sfa(o)
teJ <- teJ.sfa(o)
cbind(eff(o),te,Mode=eff(o, type="Mode"),teM,teJ)[1:10,]

sigma2.sfa(o)      # Estimated varians
lambda.sfa(o)     # Estimated lambda
```

---

 slack

---

*Calculate slack in an efficiency analysis*


---

**Description**

Slacks are calculated after taking the efficiency into consideration.

**Usage**

```
slack(X, Y, e, XREF = NULL, YREF = NULL, FRONT.IDX = NULL, LP = FALSE)
```

**Arguments**

X	Inputs of firms to be evaluated, a $K \times m$ matrix of observations of $K$ firms with $m$ inputs (firm $x$ input).
Y	Outputs of firms to be evaluated, a $K \times n$ matrix of observations of $K$ firms with $n$ outputs (firm $x$ input).
e	A Farrell object as returned from <a href="#">dea</a> et al.
XREF	Inputs of the firms determining the technology, defaults to X
YREF	Outputs of the firms determining the technology, defaults to Y
FRONT .IDX	Index for firms determining the technology
LP	Set TRUE for debugging.

**Details**

Slacks are calculated in a LP problem where the sum of all slacks are maximied after correction for efficiency. The for calculating slacks for orientation graph is low because of the low precision in the calculated graph efficiency.

**Value**

The result is returned as the Farrell object used as the argument in the call of the function with the following added components:

slack	A logical vector where the component for a firm is TRUE if the sums of slacks for the corresponding firm is positive. Only calculated in <code>dea</code> when option <code>SLACK=TRUE</code>
sum	A vector with sums of the slacks for each firm. Only calculated in <code>dea</code> when option <code>SLACK=TRUE</code>
sx	A matrix for input slacks for each firm, only calculated if the option <code>SLACK</code> is TRUE or returned from the method <code>slack</code>
sy	A matrix for output slack, see <code>sx</code>

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**References**

Peter Bogetoft and Lars Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011. Sect. 5.6 page 127.

WW Cooper, LM Seiford, and K Tone; *Data Envelopment Analysis: A Comprehensive Text with Models, Applications, References and DEA-Solver Software*, 2nd edn. Springer 2007 .

**Examples**

```
x <- matrix(c(100,200,300,500,100,200,600),ncol=1)
y <- matrix(c(75,100,300,400,25,50,400),ncol=1)
dea.plot.frontier(x,y,txt=1:dim(x)[1])

e <- dea(x,y)
eff(e)

# calculate slacks
sl <- slack(x,y,e)
data.frame(e$eff,sl$slack,sl$sx,sl$sy)
```

---

typeIerror

---

*Probability of type I error for test in a bootstrap DEA model*


---

**Description**

Calculates the probability of a type I error for a test in bootstrapped DEA models.

**Usage**

```
typeIerror(shat,s)
```

**Arguments**

shat	The value of the statistic for which the probability of a type I error is to be calculated
s	Vector with calculated values of the statistic for each of the NREP bootstraps; NREP is from dea.boot

**Details**

Needs bootstrapped values of the test statistic

**Value**

Returns the probability of a type I error

**Author(s)**

Peter Bogetoft and Lars Otto <larsot23@gmail.com>

**See Also**

boot.sw98 in **FEAR**, Paul W. Wilson (2008), “FEAR 1.0: A Software Package for Frontier Efficiency Analysis with R,” *Socio-Economic Planning Sciences* 42, 247–254

**Examples**

```
# Probability of getting something larger than 1.96 in 10000 random
# standard normal variates.
x <- rnorm(10000)
typeIerror(1.96,x)
```

# Index

- \*Topic **DEA**
  - cost.opt, 7
- \*Topic **Malmquist**
  - malmq, 40
  - malmquist, 42
- \*Topic **\textasciitildekwd1**
  - lambda, 37
  - peers, 49
- \*Topic **\textasciitildekwd2**
  - lambda, 37
  - peers, 49
- \*Topic **bootstrap**
  - dea.boot, 17
- \*Topic **datasets**
  - charnes1981, 5
  - milkProd, 46
  - norWood2004, 47
  - pigdata, 51
  - projekt, 52
- \*Topic **efficiencies**
  - dea.boot, 17
  - malmq, 40
  - malmquist, 42
- \*Topic **efficiency**
  - Benchmarking-package, 2
  - cost.opt, 7
  - dea, 11
  - dea.direct, 20
  - dea.merge, 26
  - eff, efficiencies, 31
  - eladder, 34
  - excess, 36
  - mea, 44
  - sdea, 53
  - sfa, 55
  - slack, 58
- \*Topic **merger**
  - dea.merge, 26
- \*Topic **models**
  - Benchmarking-package, 2
  - cost.opt, 7
  - dea, 11
  - dea.add, 15
  - dea.direct, 20
  - dea.merge, 26
  - dea.plot, 28
  - eladder, 34
  - excess, 36
  - mea, 44
  - sdea, 53
  - slack, 58
- \*Topic **package**
  - Benchmarking-package, 2
- \*Topic **panel**
  - malmquist, 42
- \*Topic **plot**
  - dea.plot, 28
  - eladder, 34
- \*Topic **sfa**
  - sfa, 55
- \*Topic **slack**
  - slack, 58
- Benchmarking (Benchmarking-package), 2
- Benchmarking-package, 2
- boot.fear (dea.boot), 17
- charnes1981, 5
- coef.sfa (sfa), 55
- cost.opt, 7
- critValue, 10
- dea, 8, 11, 17, 21, 23, 26, 27, 29, 31–38, 40–45, 50, 53–55, 59
- dea.add, 15
- dea.boot, 17
- dea.direct, 20, 36, 50
- dea.dual, 24
- dea.merge, 26, 39

dea.plot, 28  
dea.plot.frontier, 45  
deaSlack (dea), 11  
  
eff (eff, efficiencies), 31  
eff, efficiencies, 31  
eff.dens, 33  
efficiencies (eff, efficiencies), 31  
eladder, 34  
excess, 36  
  
get.number.peers (peers), 49  
get.which.peers (peers), 49  
  
lambda, 37  
lambda.sfa (sfa), 55  
logLik.sfa (sfa), 55  
  
make.merge, 27, 38  
malmq, 40, 42, 43  
malmquist, 42  
mea, 31, 36, 44  
milkProd, 46  
  
norWood2004, 47  
  
outlier.ap, 48  
  
peers, 49  
pigdata, 51  
print.cost.opt (cost.opt), 7  
print.Farrell (dea), 11  
print.profit.opt (cost.opt), 7  
print.revenue.opt (cost.opt), 7  
print.sfa (sfa), 55  
print.slack (slack), 58  
profit.opt (cost.opt), 7  
projekt, 52  
  
residuals.sfa (sfa), 55  
revenue.opt (cost.opt), 7  
  
sdea, 31, 53  
sfa, 31, 32, 55  
sigma2.sfa (sfa), 55  
sigma2u.sfa (sfa), 55  
sigma2v.sfa (sfa), 55  
slack, 12, 14, 21, 50, 58  
summary.cost.opt (cost.opt), 7  
summary.Farrell (dea), 11  
summary.profit.opt (cost.opt), 7  
summary.revenue.opt (cost.opt), 7  
summary.sfa (sfa), 55  
summary.slack (slack), 58  
  
te.add.sfa (sfa), 55  
te.sfa (sfa), 55  
teBC.sfa (sfa), 55  
teJ.sfa (sfa), 55  
teMode.sfa (sfa), 55  
TypeError, 60  
  
ucminf, 57