Package 'DEA'

September 10, 2026

Version 1.0.0

Date 2026-08-28

Title Data Envelopment Analysis

Type Package

Maintainer David Bernstein <davebernstein1@gmail.com>

Description Nonparametric efficiency measurement by data envelopment analysis. Provides radial (Charnes-Cooper-Rhodes and Banker-Charnes-Cooper) technical efficiency under constant, variable, non-increasing and non-decreasing returns to scale, the slacks-based measure of Tone (2001), the additive model of Charnes and others (1985), and the directional distance function of Chambers, Chung and Fare (1996), all through one interface and one result object. Efficiency estimates are accompanied by peers, slacks, returns-to-scale classification, scale efficiency and the optimal multipliers, and by bias-corrected estimates and confidence intervals from the smoothed homogeneous bootstrap of Simar and Wilson (1998). Where prices are known, cost, revenue and Nerlovian profit efficiency separate the technical component from the allocative one; where they are not, cross-efficiency with the secondary goals of Doyle and Green (1994) ranks units that a self-appraisal leaves tied. This package succeeds the archived 'DEA' package of Diaz-Martinez and Fernandez-Menendez (2008).

Suggests knitr, rmarkdown, testthat (>= 3.0.0), Benchmarking, DJL, parallel, grDevices

Imports lpSolveAPI, stats, graphics, utils

Depends R (>= 4.0.0)

License GPL (>= 2)

Language en-US

URL https://www.davidharrybernstein.com/software, https://github.com/davidhbernstein/DEA

BugReports https://github.com/davidhbernstein/DEA/issues

LazyLoad yes

LazyData true

NeedsCompilation no

VignetteBuilder knitr

Config/testthat/edition 3

Author David Bernstein [aut, cre] (ORCID:
<https://orcid.org/0000-0002-2267-5741>),
Zuleyka Diaz-Martinez [aut],
Jose Fernandez-Menendez [aut]

Repository CRAN

Date/Publication 2026-09-10 08:40:02 UTC

Contents

`DEA-package` *Data Envelopment Analysis*

Description

Nonparametric efficiency measurement by data envelopment analysis. Provides radial (Charnes-Cooper-Rhodes and Banker-Charnes-Cooper) technical efficiency under constant, variable, non-increasing and non-decreasing returns to scale, the slacks-based measure of Tone (2001), the additive model of Charnes and others (1985), and the directional distance function of Chambers, Chung and Fare (1996), all through one interface and one result object. Efficiency estimates are accompanied by peers, slacks, returns-to-scale classification, scale efficiency and the optimal multipliers, and by bias-corrected estimates and confidence intervals from the smoothed homogeneous bootstrap of Simar and Wilson (1998). Where prices are known, cost, revenue and Nerlovian profit efficiency separate the technical component from the allocative one; where they are not, cross-efficiency with the secondary goals of Doyle and Green (1994) ranks units that a self-appraisal leaves tied. This package succeeds the archived 'DEA' package of Diaz-Martinez and Fernandez-Menendez (2008).

Details

Technical efficiency – five estimators, one interface and one result object:

`dea` radial Debreu-Farrell efficiency under constant, variable, non-increasing or non-decreasing returns, or free disposal, in either orientation, with second-stage slacks and super-efficiency.

`dea_sbm` the slacks-based measure of Tone (2001).

`dea_ddf` the directional distance function of Chambers, Chung and Fare (1996), which handles zero and negative data.

`dea_add` the additive model of Charnes et al. (1985), unweighted or as the Range Adjusted Measure.

`dea_rts` scale efficiency and the returns-to-scale classification.

When prices are known, efficiency splits into a technical part and a part that is about buying the wrong mix at those prices:

`dea_cost`, `dea_revenue` cost and revenue efficiency, each factoring exactly into technical times allocative.

`dea_profit` Nerlovian profit inefficiency, which adds rather than multiplies and ties back to the directional distance function.

When they are not, and the efficient set is too large to rank:

`dea_cross` cross-efficiency – peer appraisal rather than self-appraisal – with the Doyle-Green secondary goals, which bracket an answer that is otherwise not reproducible across solvers.

`multipliers` the optimal weights themselves, from the multiplier form of any radial fit.

and, because a DEA score is an estimate and not a measurement,

`dea_boot` bias correction and confidence intervals by the smoothed homogeneous bootstrap of Simar and Wilson (1998).

`dea_rate` the rate at which any of this can converge, which depends on the number of inputs and outputs and is never root-n.

`dea_sim` a technology whose efficiency is known in closed form, for checking an estimator against a truth rather than against another implementation.

The DESCRIPTION file:

Package:	DEA
Version:	1.0.0
Date:	2026-08-28
Title:	Data Envelopment Analysis
Type:	Package
Authors@R:	c(person("David", "Bernstein", email = "davebernstein1@gmail.com", role = c("aut", "cre"), comme
Maintainer:	David Bernstein <davebernstein1@gmail.com>
Description:	Nonparametric efficiency measurement by data envelopment analysis. Provides radial (Charnes-Coo
Suggests:	knitr, rmarkdown, testthat (>= 3.0.0), Benchmarking, DJL, parallel, grDevices
Imports:	lpSolveAPI, stats, graphics, utils
Depends:	R (>= 4.0.0)

License:	GPL (>= 2)
Language:	en-US
URL:	https://www.davidharrybernstein.com/software, https://github.com/davidhbernstein/DEA
BugReports:	https://github.com/davidhbernstein/DEA/issues
LazyLoad:	yes
LazyData:	true
NeedsCompilation:	no
VignetteBuilder:	knitr
Config/testthat/edition:	3
Author:	David Bernstein [aut, cre] (ORCID: <https://orcid.org/0000-0002-2267-5741>), Zuleyka Diaz-Marti

Index of help topics:

Author(s)

David Bernstein [aut, cre] (ORCID: <https://orcid.org/0000-0002-2267-5741>), Zuleyka Diaz-Martinez [aut], Jose Fernandez-Menendez [aut]

See Also

https://www.davidharrybernstein.com/software

https://github.com/davidhbernstein/DEA

Examples

```
sim <- dea_sim(100, p = 2, q = 1, seed = 1)

fit <- dea(sim$x, sim$y, rts = "vrs", orientation = "in")
summary(fit)

dea_sbm(sim$x, sim$y, rts = "vrs")
```

```
dea_ddf(sim$x, sim$y, direction = "both", rts = "vrs")
dea_rts(sim$x, sim$y)

## With prices: on the frontier, but at the wrong point on it?
dea_cost(sim$x, sim$y, w = c(1.2, 0.8), rts = "crs")

## Without prices: ranking the units that all scored 1.
dea_cross(sim$x, sim$y, secondary = "benevolent")
```

`charnes1981` *Program Follow Through: the original data envelopment analysis data set*

Description

Seventy US primary school sites from Program Follow Through, a federally sponsored programme of remedial assistance to disadvantaged pupils. This is the data set on which Charnes, Cooper and Rhodes introduced and applied the model that carries their name, and it remains the standard worked example in the field.

Usage

```
data(charnes1981)
```

Format

A data frame with 70 rows and 10 columns.

`site` School site name. Not unique: 38 distinct names across 70 rows, since several cities contributed more than one site — New York and Philadelphia nine each.

`pft` Logical. `TRUE` for the 49 sites enrolled in Program Follow Through (rows 1–49), `FALSE` for the 21 that were not (rows 50–70).

`x1` Input: education level of the mother.

`x2` Input: highest occupation of a family member.

`x3` Input: parental visits to the school.

`x4` Input: time spent with children on school-related topics.

`x5` Input: number of teachers at the site.

`y1` Output: reading score.

`y2` Output: mathematics score.

`y3` Output: self-esteem score.

Details

Five inputs and three outputs on seventy units, which by the rule of thumb $n \geq 3(p+q)$ is comfortably enough — 70 against 24 — and is nonetheless a problem in eight dimensions. Expect a large efficient set, and see the discussion of the curse of dimensionality in `dea_rate` before reading much into the scores of the units that reach 1.

The `pft` flag makes this a natural example for an explicit reference technology: scoring the enrolled sites against the frontier spanned by the non-enrolled ones, or the reverse, is a metafrontier comparison and is what `xref`/`yref` are for.

Note

The site names are as distributed, including the misspellings "Chigago" and "Philidelphia". They are left alone rather than silently corrected, so that this copy can be compared against the others without a diff appearing where none belongs.

Source

Charnes, A., Cooper, W. W. and Rhodes, E. (1981). Evaluating program and managerial efficiency: an application of data envelopment analysis to Program Follow Through. *Management Science* 27, 668–697.

The values here are identical to those distributed as `Benchmarking::charnes1981` and `npsf::ccr81`; the two were checked against each other before this copy was made and agree on every one of the 70×8 measurements. The columns have been renamed and the programme indicator turned into a logical, but no number has been altered.

References

Charnes, A., Cooper, W. W. and Rhodes, E. (1978). Measuring the efficiency of decision making units. *European Journal of Operational Research* 2, 429–444.

Examples

```
data(charnes1981)
x <- charnes1981[, c("x1", "x2", "x3", "x4", "x5")]
y <- charnes1981[, c("y1", "y2", "y3")]

## The model of the original paper: constant returns, input oriented.
ccr <- dea(x, y, rts = "crs", orientation = "in")
ccr

## Variable returns, and the scale decomposition between the two.
dea_rts(x, y, orientation = "in")


## Score the enrolled sites against the frontier spanned by the sites that
## were NOT enrolled -- a metafrontier comparison, not an internal ranking.
inp <- charnes1981$pft
dea(x[inp, ], y[inp, ], rts = "vrs", orientation = "in",
    xref = x[!inp, ], yref = y[!inp, ])
```

dea *Radial technical efficiency*

Description

Debreu-Farrell radial efficiency by data envelopment analysis, under any of five returns-to-scale assumptions and either orientation, with optional second-stage slacks and Andersen-Petersen super-efficiency.

Usage

```
dea(x, y, data = NULL,
    rts = c("vrs", "crs", "nirs", "ndrs", "fdh"),
    orientation = c("in", "out"),
    slack = TRUE, super = FALSE, multipliers = FALSE,
    peers = TRUE, scaling = TRUE,
    xref = NULL, yref = NULL, dataref = NULL)
```

Arguments

`x`	Inputs, one row per decision-making unit (DMU) and one column per input. A matrix, a data frame, or a numeric vector for a single input. With `data` supplied it may instead be a character vector of column names or a one-sided formula such as `~ labour + capital`.
`y`	Outputs, in the same forms as `x`.
`data`	Optional data frame holding the columns named by `x` and `y`.
`rts`	Returns to scale, i.e. the restriction placed on the intensity weights: `"vrs"` variable, $\sum \lambda = 1$; the BCC model of Banker, Charnes and Cooper (1984). The default. `"crs"` constant, λ unrestricted; the CCR model of Charnes, Cooper and Rhodes (1978). `"nirs"` non-increasing, $\sum \lambda \leq 1$. `"ndrs"` non-decreasing, $\sum \lambda \geq 1$. `"fdh"` free disposal hull, λ binary; drops convexity altogether (Deprins, Simar and Tulkens 1984). Matching ignores case, and `"drs"`/`"irs"` are accepted as aliases for `"nirs"`/`"ndrs"`, which is how **Benchmarking** spells them.
`orientation`	`"in"` contracts inputs at fixed output and returns $\theta \in (0, 1]$; `"out"` expands outputs at fixed input and returns $\phi \geq 1$. Both equal 1 on the frontier. Use `efficiency(fit, "score")` to put either on a common $(0, 1]$ scale.
`slack`	Run the second-stage slack-maximizing program. Its cost is small – the constraint matrix does not depend on the DMU – and without it the `efficient` component reports radial efficiency rather than Pareto-Koopmans efficiency. See 'Details'.

`super` Andersen-Petersen (1993) super-efficiency: score each DMU against a technology it is excluded from, which breaks the ties among the efficient units. See 'Details' for the infeasibility that comes with it.

`multipliers` Also solve the *multiplier* (dual) program at each DMU and return the optimal weights v on the inputs, u on the outputs and the returns-to-scale intercept u_0. Off by default because it is a second linear program per DMU and most callers want the score, which strong duality makes identical either way. Not available for `rts = "fdh"`, which is not convex and so has no multiplier form. See 'Details' and `multipliers`.

`peers` Keep the $n \times n$ matrix of intensity weights. Set `FALSE` for large samples, where it is the dominant memory cost.

`scaling` Rescale every input and output column to mean one before solving. Radial efficiency is invariant to the units of each variable, so this changes no answer and conditions the linear program considerably better; slacks are returned in the original units either way.

`xref, yref` An explicit reference technology: the DMUs that SPAN the frontier, where `x` and `y` are the DMUs SCORED against it. Both must be given together. Default `NULL` uses `x` and `y` themselves, which is ordinary DEA.

`dataref` Data frame for `xref`/`yref` when those are given as names or formulas. Defaults to `data`.

Details

The radial score is not the whole story. A DMU projected onto a vertical or horizontal face of the frontier is radially efficient and still dominated: some input can be cut further without giving up any output. The second stage maximizes total slack at the radial projection, and `efficient` combines the two into the Pareto-Koopmans judgement. Where the two differ, `print()` says so. `dea_sbm` builds the same judgement into a single number.

Peer sets are not unique. These linear programs frequently have multiple optima, so `lambda` is one valid set of benchmarks rather than the set. Efficiency scores are unique; the weights that achieve them generally are not, and no DEA package can make them so.

Super-efficiency and infeasibility. Removing a DMU from its own reference set can leave the program with no solution at all: under `"vrs"`, `"nirs"` or `"ndrs"` a DMU at the edge of the input space may have no other unit able to dominate it. `eff` is `NA` there. That is the correct answer and not a numerical failure – a package that returns a large finite number instead is reporting something the model does not define. Infeasibility cannot occur for input-oriented `"crs"`.

The curse of dimensionality. DEA is consistent but converges slowly, at a rate that worsens with every variable added: see `dea_rate`. With $n < 3(p + q)$ most DMUs are efficient by dimension alone and `dea()` warns.

Sampling variation. Because the frontier is spanned by the observed DMUs it lies inside the true one, so every score is biased toward 1. See `dea_boot` for the bias correction and confidence intervals.

Value

An object of class `"dea"`: a list whose components are

`eff`	Named numeric vector of efficiency scores, on the orientation's own scale.
`model`, `rts`, `orientation`, `super`	The specification, as resolved.
`lambda`	Matrix of intensity weights, evaluated DMUs by reference DMUs, or `NULL` when `peers = FALSE`.
`sum_lambda`	Row sums of `lambda` from the first-stage program.
`slack_x`, `slack_y`	Input and output slacks in the original units, or `NULL` when `slack = FALSE`.
`efficient`	Logical. Pareto-Koopmans efficiency when slacks were computed – on the frontier radially AND with no slack left – and radial efficiency otherwise.
`status`	Per-DMU **lpSolveAPI** return code; 0 is an optimal solve.
`x`, `y`, `xref`, `yref`	The data, as coerced.
`n`, `nref`, `p`, `q`	Counts of evaluated DMUs, reference DMUs, inputs and outputs.
`total_time`	Elapsed seconds.
`call`	The matched call.

References

Andersen, P. and Petersen, N. C. (1993). A procedure for ranking efficient units in data envelopment analysis. *Management Science* 39, 1261–1264.

Banker, R. D., Charnes, A. and Cooper, W. W. (1984). Some models for estimating technical and scale inefficiencies in data envelopment analysis. *Management Science* 30, 1078–1092.

Charnes, A., Cooper, W. W. and Rhodes, E. (1978). Measuring the efficiency of decision making units. *European Journal of Operational Research* 2, 429–444.

Deprins, D., Simar, L. and Tulkens, H. (1984). Measuring labor-efficiency in post offices. In M. Marchand, P. Pestieau and H. Tulkens (eds), *The Performance of Public Enterprises*. North-Holland.

See Also

`dea_sbm`, `dea_ddf`, `dea_rts`, `dea_boot`, `efficiency`, `dea_sim`

Examples

```
sim <- dea_sim(60, p = 2, q = 1, seed = 1)

fit <- dea(sim$x, sim$y, rts = "vrs", orientation = "in")
fit
head(efficiency(fit))
head(peers(fit))

## The same call from a data frame, naming columns.
d <- sim$data
dea(x = c("x1", "x2"), y = "y1", data = d, rts = "crs")

## Score one group against another group's frontier.
dea(sim$x[1:10, ], sim$y[1:10, , drop = FALSE], rts = "vrs",
    xref = sim$x[11:60, ], yref = sim$y[11:60, , drop = FALSE])
```

dea-methods *Methods for objects of class "dea"*

Description

Extractors and display methods shared by dea, dea_sbm and dea_ddf, which all return the same class.

Usage

```
efficiency(object, ...)
## S3 method for class 'dea'
efficiency(object, type = c("natural", "score"), ...)

peers(object, ...)
## S3 method for class 'dea'
peers(object, threshold = 1e-08, ...)

slacks(object, ...)
## S3 method for class 'dea'
slacks(object, ...)

multipliers(object, ...)
## S3 method for class 'dea'
multipliers(object, ...)
## S3 method for class 'dea_cross'
multipliers(object, ...)

## S3 method for class 'dea'
print(x, ...)
## S3 method for class 'dea'
summary(object, ...)
## S3 method for class 'dea'
plot(x, ngrid = 400, ...)
## S3 method for class 'dea'
fitted(object, ...)
## S3 method for class 'dea'
nobs(object, ...)
```

Arguments

object, x	An object of class "dea", or – for multipliers() – one of class "dea_cross".
type	"natural" returns the model's own measure — θ, ϕ, ρ or β. "score" maps all of them onto $(0, 1]$ with 1 meaning efficient, which is what a table comparing models needs.
threshold	Intensity weights at or below this are not reported as peers.

`ngrid`	Number of grid points used to trace the frontier in `plot()`.
`...`	Passed to the underlying plot, or ignored.

Details

`efficiency(object, "score")` deliberately *fails* for a directional fit whose direction is neither purely input nor purely output. β is an additive distance measured in units of g and there is no ratio it corresponds to; a formula such as $(1-\beta)/(1+\beta)$ would produce a number that looks comparable to a Farrell score and is not.

`fitted()` returns the frontier projection: where each DMU would sit if it were efficient. For a radial fit with slacks that is the radial projection with the remaining slack removed, so the result is a Pareto-Koopmans point.

`plot()` draws the estimated frontier when there is one input and one output, and the distribution of scores otherwise. The frontier is traced by *evaluating* the fitted technology on a grid of input levels rather than by joining up the efficient DMUs, so the same code draws it correctly under every returns-to-scale assumption — including the free disposal hull's staircase — and puts the vertical and horizontal faces where they actually are.

Peer sets are not unique. These linear programs frequently have multiple optima, so `peers()` reports one valid benchmark set rather than the benchmark set. Efficiency scores are unique; the weights achieving them generally are not.

Value

`efficiency` and `nobs` return numeric vectors; `peers` returns a data frame with columns `dmu`, `peer` and `lambda`; `slacks` and `fitted` return matrices with one row per DMU. `print` and `summary` return their argument invisibly.

The multiplier form

`multipliers()` returns one row per DMU: the weights v on the inputs, u on the outputs, and u_0 where the technology has one. They are the prices that make the DMU look as good as it possibly can while valuing no reference DMU above break-even, and they satisfy $v'x_o = 1$ (input orientation) in the caller's own units.

For an efficient DMU they are not unique. An efficient unit generally has a whole face of optimal weight vectors, every one giving it the same score of 1 and giving other DMUs different ones, so what comes back is one vertex of that face and another solver may return a different vertex. The score is unique; the prices supporting it are not. `dea_cross` exists largely to deal with the consequences.

See Also

`dea`, `dea_sbm`, `dea_ddf`

Examples

```
sim <- dea_sim(60, p = 1, q = 1, seed = 1)
fit <- dea(sim$x, sim$y, rts = "vrs")
```

```
summary(fit)
head(efficiency(fit, "score"))
head(peers(fit))
head(slacks(fit))
head(fitted(fit))

plot(fit)
```

---

dea_add *Additive efficiency, and the weighted measures built on it*

Description

The additive model of Charnes, Cooper, Golany, Seiford and Stutz (1985): maximize total slack over the technology, with no orientation and no radial contraction. Available unweighted, or normalized as the Range Adjusted Measure or the Measure of Inefficiency Proportions.

Usage

```
dea_add(x, y, data = NULL,
        measure = c("ram", "mip", "unweighted"),
        rts = c("vrs", "crs", "nirs", "ndrs"),
        peers = TRUE, scaling = TRUE,
        xref = NULL, yref = NULL, dataref = NULL)
```

Arguments

x, y, data	Inputs and outputs, in any of the forms accepted by dea. measure = "mip" additionally requires every value to be strictly positive, because it divides each slack by the DMU's own level.
measure	Which weights the objective carries:
	"ram" Range Adjusted Measure (Cooper, Park and Pastor 1999). Each slack is divided by the range of its variable across the reference set, and by $p + q$. Units invariant, and the reported score $\rho = 1 -$ objective lies in $[0, 1]$, because no slack can exceed its variable's range. The default.
	"mip" Measure of Inefficiency Proportions. Each slack is divided by the evaluated DMU's own level of that variable, and by $p + q$. Units invariant, and at most 1 — but **not bounded below by 0**: an input slack cannot exceed the DMU's own input, so its term is capped, while an output slack has no such ceiling and a badly performing DMU can score negative. RAM is bounded on both sides, which is what it was designed for, and is the better default.
	"unweighted" The original 1985 model. Reported as the raw slack total, so **0 is efficient and larger is worse** — and it carries the units of the data. See 'Details'.
rts	Returns to scale, as in dea. There is no "fdh" option — the measure is defined on a convex technology.

`peers`, `scaling`, `xref`, `yref`, `dataref`
: As in `dea`, except that `scaling` is ignored for `measure = "unweighted"`: that measure is defined on the caller's units, so rescaling the columns would change the answer rather than leave it alone.

Details

One model, three scales. All three options solve the same program over the same technology and differ only in the objective weights. The literature often presents RAM and MIP as separate models; they are one model read three ways, and the projection and the efficient set are identical across them.

The classic version is not units invariant, and that is why the other two exist. With unit weights the objective adds a slack measured in staff-hours to one measured in euros: double an input's units and its slack halves, which can change which DMU looks worst. The efficient *set* is unaffected — a slack is zero in any units — but the *ordering* of the inefficient DMUs is not. `efficiency(fit, "score")` therefore refuses to put the unweighted measure on a common scale, and `scaling = TRUE` is ignored for it.

Solving in the caller's units also means the unweighted program is poorly conditioned when the columns differ wildly in magnitude, and `dea_add()` warns above a spread of 10^3. On a design with one input multiplied by 1000, one DMU failed numerically outright. That is a real property of the measure and another reason to prefer `"ram"`.

No orientation. The additive model does not ask by what factor inputs could shrink or outputs grow; it asks how far the DMU is from the frontier in every coordinate at once. That makes it the natural tool when the question is *which* units are dominated rather than *by how much*, and it is why its efficient set coincides exactly with `dea_sbm`'s $\rho = 1$ set and with `dea`'s Pareto-Koopmans set.

Infeasibility with an external reference set. As for `dea_sbm`, the balance rows are equalities with non-negative slacks, so a solution exists only if some convex combination of the reference DMUs weakly dominates the evaluated point in every input and output. A point outside the estimated technology has none; `dea_add()` warns and reports `NA`. For the same reason `"ram"` can return a score below 0 when scoring out of sample, since a slack may then exceed the reference range.

Value

An object of class `"dea"` with `model = "additive"` and the chosen `measure` recorded. `efficient` is the Pareto-Koopmans efficient set on every scale.

References

Charnes, A., Cooper, W. W., Golany, B., Seiford, L. and Stutz, J. (1985). Foundations of data envelopment analysis for Pareto-Koopmans efficient empirical production functions. *Journal of Econometrics* 30, 91–107.

Cooper, W. W., Park, K. S. and Pastor, J. T. (1999). RAM: a range adjusted measure of inefficiency for use with additive models, and relations to other models and measures in DEA. *Journal of Productivity Analysis* 11, 5–42.

See Also

`dea_sbm`, `dea`, `dea_ddf`

Examples

```
data(charnes1981)
x <- charnes1981[, paste0("x", 1:5)]
y <- charnes1981[, paste0("y", 1:3)]

dea_add(x, y, measure = "ram", rts = "vrs")

## The efficient set is the Pareto-Koopmans set, identical to the one the
## radial model finds after its second stage and the one dea_sbm() scores 1.
a  <- dea_add(x, y, measure = "ram", rts = "vrs")
r  <- dea(x, y, rts = "vrs", orientation = "in", slack = TRUE)
identical(unname(a$efficient), unname(r$efficient))

## RAM is units invariant; the unweighted model is not.
x2 <- x; x2[, 1] <- x2[, 1] * 1000
max(abs(dea_add(x2, y, measure = "ram")$eff - dea_add(x, y, measure = "ram")$eff))
max(abs(dea_add(x2, y, measure = "unweighted")$eff -
        dea_add(x, y, measure = "unweighted")$eff))
```

---

dea_boot *Bias correction and confidence intervals by the Simar-Wilson bootstrap*

Description

The smoothed homogeneous bootstrap of Simar and Wilson (1998): bias-corrected efficiency estimates and confidence intervals for a radial DEA fit.

Usage

```
dea_boot(object, B = 2000, alpha = 0.05,
         bw = c("silverman", "nrd0", "ucv", "sj"),
         seed = NULL, ncores = 1L, progress = interactive())

## S3 method for class 'dea_boot'
print(x, ...)
## S3 method for class 'dea_boot'
summary(object, ...)
```

Arguments

object	A fit from dea, scored against its own sample (no xref), not a super-efficiency or free-disposal-hull fit. For the print method, an object of class "dea_boot".
B	Bootstrap replications. 2000 is the usual recommendation; each one costs a full DEA sweep, so the run is B times the cost of the fit.
alpha	One minus the nominal coverage of the interval.

bw	Bandwidth for the kernel smoothing, as a rule name or a positive number. "silverman" is the robust normal-reference rule Simar and Wilson use, applied to the reflected sample; the others are bw.nrd0, bw.ucv and bw.SJ.
seed	Integer seed. The caller's random-number stream is restored afterwards.
ncores	Replications run in parallel through parallel::mclapply when this exceeds 1. Not available on Windows; a request there runs serially with a message.
progress	Print a counter while running serially.
x	An object of class "dea_boot".
...	Ignored.

Details

Why it is needed. The DEA frontier is spanned by the observed DMUs, so it lies inside the true one: every efficiency score is biased toward 1, and the bias is not small. A DEA score reported on its own is a point estimate of unknown accuracy from an estimator with a known, one-signed bias. This is the largest gap between how DEA is used in practice and what is known about it.

Smoothing and reflection. Resampling the estimated efficiencies directly is inconsistent: it puts an atom of probability on each observed score, in particular a large one on 1, so the pseudo-frontier is a fixed set of points rather than a draw from a density. Smoothing with a Gaussian kernel fixes that but would leak probability past the boundary at 1, exactly where the mass is. The remedy is reflection: smooth the $2n$-point set $\{\hat{\theta}_i\} \cup \{2 - \hat{\theta}_i\}$, which is symmetric about 1 by construction, then fold draws back across it.

Whether to use the correction. bias_corrected is $2\hat{\theta} - \mathrm{mean}(\hat{\theta}^*)$. Simar and Wilson warn against applying it unconditionally: it removes a bias but adds the variance of the estimate of that bias, and is a net loss unless the bias is large relative to the bootstrap standard error. Their rule of thumb — correct only when $|bias|/se > 1/\sqrt{3}$ — is evaluated per DMU and reported as correct_worthwhile, rather than being applied silently.

Intervals. The interval is the percentile interval for $\hat{\theta} - \theta$, inverted, and so is centred near bias_corrected rather than near eff.

Value

An object of class "dea_boot". Its table component is a data frame with one row per DMU and columns dmu, eff, bias, se, bias_corrected, ci_lower, ci_upper and correct_worthwhile. The full n by B matrix of bootstrap scores is returned as boot, and the bandwidth actually used as bw.

References

Simar, L. and Wilson, P. W. (1998). Sensitivity analysis of efficiency scores: how to bootstrap in nonparametric frontier models. *Management Science* 44, 49–61.

Simar, L. and Wilson, P. W. (2000). A general methodology for bootstrapping in nonparametric frontier models. *Journal of Applied Statistics* 27, 779–802.

See Also

dea, dea_rate

Examples

```
sim <- dea_sim(40, p = 1, q = 1, seed = 3)
fit <- dea(sim$x, sim$y, rts = "vrs", orientation = "in")

## B is tiny here so the example runs quickly; use 2000 in practice.
b <- dea_boot(fit, B = 50, seed = 1, progress = FALSE)
b
head(b$table)

## The correction moves the estimates toward the truth.
c(raw = mean(fit$eff - sim$theta),
  corrected = mean(b$table$bias_corrected - sim$theta))
```

---

dea_cost *Cost, revenue and profit efficiency*

Description

Efficiency when prices are known. Technical efficiency asks whether a DMU is on the frontier; these ask whether it is at the right *point* on it. A DMU can be perfectly efficient technically and still be buying the wrong input mix for the prices it faces, and only a price-based measure sees that.

Usage

```
dea_cost(x, y, w, data = NULL,
         rts = c("vrs", "crs", "nirs", "ndrs"),
         peers = TRUE, scaling = TRUE,
         xref = NULL, yref = NULL, dataref = NULL)

dea_revenue(x, y, r, data = NULL,
            rts = c("vrs", "crs", "nirs", "ndrs"),
            peers = TRUE, scaling = TRUE,
            xref = NULL, yref = NULL, dataref = NULL)

dea_profit(x, y, w, r, data = NULL,
           direction = c("both", "in", "out", "unit", "mean"),
           rts = c("vrs", "nirs"),
           peers = TRUE, scaling = TRUE,
           xref = NULL, yref = NULL, dataref = NULL)
```

Arguments

x, y, data	Inputs and outputs, in any of the forms accepted by dea.
w	Input prices. Either a length-p numeric vector, meaning one price list faced by every DMU, or an $n \times p$ matrix or data frame with one row per DMU, or – with data – column names or a one-sided formula. All prices must be strictly positive. A bare vector is read as a shared price list, *not* as one observation per

DMU; when $n = p$ makes the two readings indistinguishable, the call is refused rather than guessed.

`r` Output prices, in the same forms, with q in place of p.

`rts` Returns to scale, as in `dea`. `dea_profit()` accepts only `"vrs"` and `"nirs"`; see **Details**.

`direction` For `dea_profit()`, the direction g the profit gap is normalized by, in the forms `dea_ddf` accepts.

`peers`, `scaling`, `xref`, `yref`, `dataref`
As in `dea`.

Details

The decomposition is the point. Cost efficiency is

$$CE_o = \frac{\min\{w_o' x : (x, y_o) \in T\}}{w_o' x_o} = \theta_o \times AE_o$$

the first factor being the input-oriented radial score and the second the loss from using the wrong mix at these prices. Both are in $(0, 1]$, and a DMU is cost efficient only if it is both technically and allocatively efficient. Revenue efficiency decomposes the same way against the output-oriented score.

Profit is not a ratio, and this implementation does not pretend it is. Observed profit is routinely zero and sometimes negative, so Π_o / Π^* is undefined or meaningless exactly where the question matters. The measure used is the Nerlovian one, the profit gap normalized by the value of a direction,

$$\frac{\Pi^* - \Pi_o}{w_o' g_x + r_o' g_y}$$

which the Chambers-Chung-Fare duality splits exactly into the directional distance β in the same direction plus an allocative remainder. Both parts are non-negative and bigger is worse, matching `dea_ddf` and reversing the convention of the other two functions.

Constant and non-decreasing returns are refused for profit, and this is a property of the problem rather than a gap in the implementation: over a cone, one profitable reference DMU makes maximum profit unbounded, because scaling that DMU up scales its profit up with it. Maximum profit is a variable-returns concept.

An external reference set can make the cost program infeasible under variable returns, for the same reason it can for `dea_sbm`: the program must match the evaluated DMU's output from a convex combination of the reference DMUs, and an outsider producing more than any of them cannot be matched. `NA` is returned with a warning naming the cause. Under constant returns the cone always reaches, so this does not arise.

Prices are assumed exogenous to the DMU. Nothing here tests that, and where a DMU is large enough to move its own prices the measures answer a question about a price-taker that the DMU is not.

Value

An object of class `"dea_price"`, a list whose components include:

`eff` Overall efficiency. For cost and revenue, a ratio in $(0, 1]$ where 1 is best. For profit, the Nerlovian gap, where **0 is best and larger is worse**.

`technical`, `allocative`
The two factors of `eff`. Cost and revenue efficiency *multiply*: `eff == technical * allocative`. The Nerlovian gap *adds*: `eff == technical + allocative`.

`optimal`, `observed`
Minimum cost and observed cost, or maximum revenue and observed revenue. Not present for `dea_profit()`, which reports `profit_max` and `profit_obs` instead.

`optimal_q` The cost-minimizing input mix, or the revenue-maximizing output mix, in the caller's units. Also returned by `fitted()`.

`lambda`, `status`, `dmu`, `ref`, `n`, `nref`, `p`, `q`
As in `dea`.

References

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

Chambers, R. G., Chung, Y. and Fare, R. (1998). Profit, directional distance functions, and Nerlovian efficiency. *Journal of Optimization Theory and Applications* 98, 351–364.

See Also

`dea`, `dea_ddf`, `dea_cross`

Examples

```
X <- as.matrix(charnes1981[, paste0("x", 1:5)])
Y <- as.matrix(charnes1981[, paste0("y", 1:3)])

## One price list faced by every school site.
ce <- dea_cost(X, Y, w = c(1.4, 0.9, 2.1, 1.2, 1.0), rts = "crs")
ce

## The decomposition is exact, and this is what it buys: sites that are on the
## frontier but are not buying the cheapest mix that would keep them there.
sum(ce$technical > 1 - 1e-9)      # technically efficient
sum(ce$eff == 1)                  # AND allocatively efficient

## Profit efficiency needs prices on both sides, and variable returns.
pe <- dea_profit(X, Y, w = c(1.4, 0.9, 2.1, 1.2, 1.0), r = c(3, 2.5, 1.8))
all.equal(unname(pe$eff), unname(pe$technical + pe$allocative))
```

`dea_cross` *Cross-efficiency*

Description

Peer appraisal instead of self-appraisal. Ordinary DEA lets every DMU pick the weights that flatter it most, which is why so many of them score 1 and why the ranking is so often empty. Cross-efficiency scores each DMU under *every other* DMU's optimal weights and averages the result.

Usage

```
dea_cross(x, y, data = NULL,
          secondary = c("benevolent", "aggressive", "none"),
          rts = c("crs", "vrs"),
          self = TRUE, scaling = TRUE,
          xref = NULL, yref = NULL, dataref = NULL)
```

Arguments

`x`, `y`, `data`	Inputs and outputs, in any of the forms accepted by `dea`.
`secondary`	How to choose among a rater's alternate optimal weights. `"benevolent"` maximizes the average cross-efficiency it awards everyone else, `"aggressive"` minimizes it, and `"none"` takes whatever the solver returned. See **Details**: `"none"` is not reproducible and is not the default.
`rts`	Returns to scale. `"vrs"` is accepted only with `secondary = "none"`.
`self`	Whether a DMU's own appraisal counts in its mean. `FALSE` drops the diagonal, which is Doyle and Green's convention.
`scaling`, `xref`, `yref`, `dataref`	As in `dea`. The reference set is the panel of *raters*; separating it from `x`/`y` lets a fixed panel score newcomers.

Details

$$E_{do} = \frac{u_d' y_o}{v_d' x_o}$$

is DMU o's efficiency priced at DMU d's own optimal weights. The column mean is o's cross-efficiency. It very rarely ties, which is the practical reason to use it: on the 70 sites of `charnes1981` the constant-returns model calls 19 of them efficient and cannot rank them, while cross-efficiency ranks all 70.

The weights are not unique, and that is the whole difficulty. An efficient DMU generally has an entire face of optimal weight vectors, all giving it the same score of 1 but giving *other* DMUs quite different cross-efficiencies. A cross-efficiency computed from whichever vertex the solver happened to stop at is therefore not reproducible across solvers, let alone across packages, and the spread is first-order rather than numerical.

Doyle and Green (1994) resolve this by holding the rater's own score at θ_d and choosing among the remaining optima with a stated secondary goal. Reporting both the benevolent and the aggressive answer brackets the result: a ranking that survives from one to the other is a property of the data, and one that does not was a property of the solver. That is why `"none"` is available but is not the default.

Constant returns is the default for a reason. Under variable returns the ratio carries the intercept u_0, is no longer bounded by 1 and can be negative, so the average of a column is not an efficiency. `dea_cross()` allows `rts = "vrs"` for description but refuses to attach a secondary goal to it, because "maximize everyone else's average" would then be optimizing something that is not an efficiency.

Value

An object of class `"dea_cross"`, a list whose components include:

`eff`	The cross-efficiency of each DMU: the mean of the appraisals it receives.
`cross_matrix`	The $n_{ref} \times n$ matrix E, one row per rater and one column per rated DMU. Its diagonal is the ordinary DEA score.
`own`	Each DMU's ordinary DEA score, or `NULL` when the rated DMUs are not among the raters.
`maverick`	Doyle and Green's index, `(own - eff) / eff`: how much a DMU's self-appraisal exceeds what its peers award it. Large means a DMU that looks efficient only under weights nobody else would choose.
`spread`	The range of appraisals each DMU receives – a direct measure of how much the answer depends on whose weights are used.
`v, u, u0`	The weights each rater used.

References

Sexton, T. R., Silkman, R. H. and Hogan, A. J. (1986). Data envelopment analysis: critique and extensions. *New Directions for Program Evaluation* 32, 73–105.

Doyle, J. and Green, R. (1994). Efficiency and cross-efficiency in DEA: derivations, meanings and uses. *Journal of the Operational Research Society* 45, 567–578.

See Also

`dea`, `multipliers`

Examples

```
X <- as.matrix(charnes1981[, paste0("x", 1:5)])
Y <- as.matrix(charnes1981[, paste0("y", 1:3)])
rownames(X) <- rownames(Y) <- charnes1981$site

ben <- dea_cross(X, Y, secondary = "benevolent")
agg <- dea_cross(X, Y, secondary = "aggressive")
summary(ben)
```

```
## The bracket. Where it is wide, the ranking is an artefact of the weights.
range(ben$eff - agg$eff)

## 19 sites tie at 1 under ordinary DEA; none tie under cross-efficiency.
sum(ben$own == 1)
length(unique(round(ben$eff, 8)))
```

---

dea_ddf *Directional distance function*

Description

The directional technology distance function of Chambers, Chung and Fare (1996): the number of units of a chosen direction vector by which a DMU could move toward the frontier, contracting inputs and expanding outputs at the same time.

Usage

```
dea_ddf(x, y, data = NULL,
        direction = c("both", "in", "out", "unit", "mean"),
        rts = c("vrs", "crs", "nirs", "ndrs"),
        peers = TRUE, scaling = TRUE,
        xref = NULL, yref = NULL, dataref = NULL)
```

Arguments

x, y, data	Inputs and outputs, in any of the forms accepted by dea. Unlike the other models here, values may be zero or negative.
direction	The direction $g = (g_x, g_y)$ along which distance is measured. One of "both" $g = (x_o, y_o)$, each DMU's own data. The default. "in" $g = (x_o, 0)$; then $\beta = 1 - \theta$. "out" $g = (0, y_o)$; then $\beta = \phi - 1$. "unit" $g = (1, \ldots, 1)$, one unit of every variable. "mean" the sample mean of each variable, the same for all DMUs. Or a numeric vector of length p + q giving one fixed direction, inputs first; or a n by p + q matrix giving one per DMU.
rts	Returns to scale, as in dea.
peers, scaling, xref, yref, dataref	As in dea.

Details

Why not just use a radial measure. Three reasons, in order of how often they bite.

Inputs and outputs move together. A radial measure has to pick a side; `"both"` contracts inputs and expands outputs at once, which is what most efficiency questions actually ask.

Zeros and negatives are fine. β is an additive contraction, so nothing is divided by a DMU's own level. A zero output leaves the radial output score undefined and the slacks-based measure undefined; here it is just a number. Net income, net exports and abatement credits can be negative, and the directional distance function is the standard way to handle them.

The direction is visible. A radial model also chooses a direction; it simply does not say which.

Units. β is scale invariant when g is proportional to the DMU's own data and is *not* when g is fixed, since doubling an input's units halves the β needed to move one unit of g. That is a property of the estimand rather than a defect, but a fixed direction must be stated in the same units as the data. Internal rescaling is applied to the direction as well, so the reported β is the one the caller's units imply.

No common ratio scale. `efficiency(fit, "score")` refuses to convert β to a $(0, 1]$ score for a direction that is not purely input or purely output, because no such conversion exists.

Value

An object of class `"dea"` with `model = "ddf"`. Note that `eff` (also available as `beta`) measures *inefficiency*: **0 is on the frontier and larger is worse**, the reverse of every other model in this package. The direction actually used is returned in `g`. There are no second-stage slacks – the projection is along g.

References

Chambers, R. G., Chung, Y. and Fare, R. (1996). Benefit and distance functions. *Journal of Economic Theory* 70, 407–419.

Chung, Y. H., Fare, R. and Grosskopf, S. (1997). Productivity and undesirable outputs: a directional distance function approach. *Journal of Environmental Management* 51, 229–240.

See Also

`dea`, `dea_sbm`

Examples

```
sim <- dea_sim(60, p = 2, q = 1, seed = 1)

dea_ddf(sim$x, sim$y, direction = "both", rts = "vrs")

## The radial models are the two one-sided directions.
b <- dea_ddf(sim$x, sim$y, direction = "in", rts = "vrs")
t <- dea(sim$x, sim$y, orientation = "in", rts = "vrs", slack = FALSE)
max(abs(b$beta - (1 - t$eff)))

## A fixed direction: one unit of every variable.
dea_ddf(sim$x, sim$y, direction = "unit", rts = "vrs")
```

`dea_rts` *Scale efficiency and returns to scale*

Description

Fit the constant-, variable- and non-increasing-returns technologies to the same data in one sweep, and use the three scores to decompose efficiency into its technical and scale parts and to classify each DMU's local returns to scale.

Usage

```
dea_rts(x, y, data = NULL, orientation = c("in", "out"), tol = 1e-6,
        scaling = TRUE)

## S3 method for class 'dea_rts'
print(x, ...)
## S3 method for class 'dea_rts'
summary(object, ...)
```

Arguments

`x`, `y`, `data`	Inputs and outputs, in any of the forms accepted by `dea`. For the `print` method, an object of class `"dea_rts"`.
`orientation`	`"in"` or `"out"`, as in `dea`.
`tol`	How close two efficiency scores must be to count as equal. The classification is a comparison of three linear-programming optima, so it needs a tolerance; `1e-6` is loose relative to the solver and tight relative to any economically meaningful difference.
`scaling`	As in `dea`.
`object`	An object of class `"dea_rts"`.
`...`	Ignored.

Details

Scale efficiency is $SE = \theta_{CRS}/\theta_{VRS}$ under an input orientation and ϕ_{VRS}/ϕ_{CRS} under an output orientation, so it lies in $(0, 1]$ either way. It is 1 when the DMU operates at the most productive scale size, and the shortfall is the part of its distance to the constant-returns frontier that comes from being the wrong *size* rather than from being badly run.

The classification uses the non-increasing technology, which permits scaling a reference point down but not up. A DMU in the increasing-returns region is smaller than the most productive scale size, so the constant-returns frontier above it is reached by scaling its peer *down* – which the non-increasing technology permits, and there it coincides with the constant-returns frontier. A DMU in the decreasing-returns region would need its peer scaled *up*, which it forbids, and there it coincides with the variable-returns frontier:

$$\theta_{CRS} = \theta_{VRS} \Rightarrow \text{constant returns}$$

$$\theta_{NIRS} = \theta_{CRS} \neq \theta_{VRS} \Rightarrow \text{increasing returns}$$

$$\theta_{NIRS} = \theta_{VRS} \neq \theta_{CRS} \Rightarrow \text{decreasing returns}$$

Note that the rule runs opposite to what the label suggests: it is agreement with the *constant*-returns frontier that marks the increasing-returns region.

The often-quoted alternative – read $\sum \lambda$ off the constant-returns program and call it increasing below 1, decreasing above – gives the same answer when that program has a unique optimum and an arbitrary one when it does not, which is common. `sum_lambda_crs` is reported so the two can be compared, but the verdict uses the rule above.

The orientation changes the answer, and is meant to. Returns to scale are a property of the frontier *point* a DMU is benchmarked against, not of the DMU's own coordinates. An output-oriented projection holds inputs fixed, so the verdict describes the DMU's own scale. An input-oriented projection moves inputs down, and for a badly inefficient DMU it can land below the most productive scale size even though the DMU itself sits above it — so the same DMU can be classified decreasing by the output sweep and increasing by the input one. Both are correct about their own projection. Choose the orientation that matches the decision being made, and do not read the two tables against each other row by row.

These are point estimates from an estimator with a known bias, and the classification inherits that. A formal test of constant returns needs a bootstrap; see Simar and Wilson (2002).

Value

An object of class `"dea_rts"` whose `table` component is a data frame with one row per DMU and columns `dmu`, `crs`, `vrs`, `nirs`, `scale_eff`, `rts` (one of `"irs"`, `"crs"`, `"drs"`) and `sum_lambda_crs`.

References

Coelli, T. J., Rao, D. S. P., O'Donnell, C. J. and Battese, G. E. (1998). *An Introduction to Efficiency and Productivity Analysis*. Springer.

Fare, R., Grosskopf, S. and Lovell, C. A. K. (1994). *Production Frontiers*. Cambridge University Press.

Simar, L. and Wilson, P. W. (2002). Non-parametric tests of returns to scale. *European Journal of Operational Research* 139, 115–132.

See Also

`dea`, `dea_boot`

Examples

```
sim <- dea_sim(80, p = 1, q = 1, returns = 0.7, seed = 2)
r <- dea_rts(sim$x, sim$y)
r
head(r$table)
```

dea_sbm *Slacks-based measure of efficiency*

Description

Tone's (2001) slacks-based measure: a non-radial efficiency score that is 1 if and only if the DMU is Pareto-Koopmans efficient, units invariant, and monotone in every input and output slack.

Usage

```
dea_sbm(x, y, data = NULL,
        rts = c("vrs", "crs", "nirs", "ndrs"),
        orientation = c("none", "in", "out"),
        peers = TRUE, scaling = TRUE,
        xref = NULL, yref = NULL, dataref = NULL)
```

Arguments

`x`, `y`, `data`	Inputs and outputs, in any of the forms accepted by `dea`. Every value must be strictly positive: the measure divides each slack by the DMU's own level.
`rts`	Returns to scale, as in `dea`. There is no `"fdh"` option – the measure is defined on a convex technology.
`orientation`	`"none"` (the default) scores input and output slack together, as a ratio; `"in"` scores input slack only and `"out"` output slack only. All three return a score in $(0, 1]$.
`peers`, `scaling`, `xref`, `yref`, `dataref`	As in `dea`.

Details

The non-oriented measure is

$$\rho = \frac{1 - (1/p)\sum_i s_i^- / x_{io}}{1 + (1/q)\sum_r s_r^+ / y_{ro}}$$

minimized over the technology. It answers a different question from the radial score: not "by what common factor can every input be cut" but "what fraction of each input is being wasted". The two coincide only when no slack remains at the radial projection, and $\rho \leq \theta$ always.

The fractional program is solved exactly. A ratio of linear forms is not a linear program, and the Charnes-Cooper change of variable used here – introduce $t > 0$, substitute $\Lambda = t\lambda$ and $S = ts$, and normalize the denominator to 1 – turns it into one without approximation.

The oriented measures are one-sided by construction. ρ_I sees only input waste and ρ_O only forgone output, so a DMU can score 1 on one of them while carrying slack on the other side. Use `"none"` unless the orientation is a substantive assumption about what the DMU controls.

An external reference set can make the program infeasible. The balance constraints are equalities with non-negative slacks, so they require some convex combination of the reference DMUs to weakly dominate the evaluated point in every input *and* every output. A DMU scored against its

own sample dominates itself, so this is free; a point scored against someone else's frontier may lie outside the estimated technology, and then there is no solution. `dea_sbm()` reports `NA` and warns, naming the cause. `dea` has no such problem — its radial score simply falls below 1 for a point outside the technology — so use it when scoring out of sample.

Zeros. Because every slack is divided by the DMU's own level, a zero input or output has no defined contribution. `dea_sbm()` stops rather than substituting a small number, which would make the score depend on the number chosen. Use `dea`, whose radial score tolerates zeros, or `dea_ddf`, whose additive measure never divides by the data.

Value

An object of class `"dea"`, with the same components as `dea`'s return value; `eff` holds ρ and `model` is `"sbm"`. Slacks are always computed, so `efficient` is `eff == 1`.

References

Tone, K. (2001). A slacks-based measure of efficiency in data envelopment analysis. *European Journal of Operational Research* 130, 498–509.

See Also

`dea`, `dea_ddf`

Examples

```
sim <- dea_sim(60, p = 2, q = 1, seed = 1)

rho <- dea_sbm(sim$x, sim$y, rts = "vrs")
rho

## rho <= theta, with equality exactly where the radial projection leaves no
## slack behind.
th <- dea(sim$x, sim$y, rts = "vrs", orientation = "in")
all(rho$eff <= th$eff + 1e-8)
identical(rho$eff == 1, unname(th$efficient))
```

`dea_sim` *Simulate a technology with known efficiency, and the rate it can be estimated at*

Description

`dea_sim` draws a sample from a production technology whose distance functions have a closed form, so that an estimator can be scored against a truth written down before it ran. `dea_rate` gives the rate at which the DEA estimator can converge on such a design, expressed as the slope of log mean squared error on log sample size.

Usage

```
dea_sim(n, p = 1L, q = 1L, returns = 1, ineff = c("exp", "hnorm"),
        mean_ineff = 0.3, x_range = c(1, 2), seed = NULL)

dea_rate(p, q, rts = c("vrs", "crs", "nirs", "ndrs", "fdh"),
         what = c("mse", "estimator"))

## S3 method for class 'dea_sim'
print(x, ...)
```

Arguments

`n`	Number of DMUs.
`p, q`	Numbers of inputs and outputs.
`returns`	Elasticity of scale of the true frontier, in $(0, 1]$. At 1 the technology is a cone, so a constant-returns fit is consistent; below 1 it is strictly concave and a constant-returns fit is inconsistent by construction, which makes it a useful negative control.
`ineff`	Distribution of the inefficiency term $u \geq 0$: exponential or half-normal. Both are parameterized so that $E[u]$ is `mean_ineff`, so the two designs are comparable.
`mean_ineff`	$E[u]$.
`x_range`	Support of each input, as two increasing positive numbers. Widening it downward matters for any estimand whose projection moves inputs down — see 'Details'.
`seed`	Integer seed. The caller's random-number stream is restored afterwards.
`rts`	Returns-to-scale assumption of the estimator whose rate is wanted.
`what`	`"mse"` returns the slope for mean squared error; `"estimator"` returns it for the estimator itself, which is half as steep.
`x`	An object of class `"dea_sim"`.
`...`	Ignored.

Details

The design. Inputs are uniform on `x_range`^{p}, by default $[1, 2]^p$. The frontier is the Cobb-Douglas aggregate $f(x) = \prod_j x_j^{r/p}$ with $r =$ `returns`, and the output set at x is the non-negative part of the ball of radius $f(x)$:

$$T = \{(x, y) : \|y\| \leq f(x)\}.$$

This is a genuine production technology in the sense DEA assumes – f is concave for $r \leq 1$ so T is convex, both f and the norm are monotone so T is freely disposable, and at $r = 1$ it is a cone. Setting $\|y\| = f(x)e^{-u}$ then gives

$$\phi = f(x)/\|y\| = e^{u}, \qquad \theta = e^{-u/r},$$

the second because scaling every input by θ scales f by θ^r. With $q > 1$ the direction of y within the ball is drawn uniformly and independently of u, so the output mix carries no information about

efficiency; fixing it instead would put every DMU on one ray, and a q-output problem on a single ray is a one-output problem wearing q columns.

`theta` **needs care, and** `x_range` **is why it is an argument.** `theta` is the distance to the technology as defined for *all* $x > 0$, and reaching it means scaling inputs down — for an inefficient DMU, below $\min(x)$. A variable-returns hull cannot extrapolate there, so on the default support it does not estimate `theta` consistently however large the sample: the mean squared error hits a floor and a log-log sweep produces a slope near zero. That is a property of the estimand, not a bug. Three ways out: use `phi`, since outputs expand at fixed x and stay inside the support; fit `rts = "crs"` against `theta` on a `returns = 1` design, where the technology is a cone and can be scaled without bound; or widen `x_range` downward so the sample has room beneath the points being scored. The same applies to the directional distance function's two-sided direction, which also moves inputs down.

The rate. DEA is consistent but not root-n consistent, and the rate worsens with every variable added. For the radial estimator with p inputs and q outputs,

$$|\hat{\theta} - \theta| = O_p(n^{-2/(p+q+1)})$$

under variable returns (Kneip, Park and Simar 1998), $O_p(n^{-2/(p+q)})$ under constant returns (Park, Simar and Weiner 2000), and $O_p(n^{-1/(p+q)})$ for the free disposal hull. Squaring doubles the exponent, so regressing log mean squared error on log n over a grid of sample sizes has slope $-4/(p+q+1)$, $-4/(p+q)$ and $-2/(p+q)$ respectively. That number, not -1, is the benchmark: the familiar root-n slope is reached only at $p+q=3$ under variable returns, and at $p+q=8$ the slope is $-4/9$, so a hundredfold increase in sample size buys about a factor of 8 in mean squared error where a parametric estimator would buy 100.

These are worst-case rates over a class of technologies. On a frontier that happens to be linear the estimator does better, so a slope *steeper* than the target is not a defect; only a slope materially flatter is.

Value

`dea_sim` returns an object of class `"dea_sim"`: a list with `x` and `y` (matrices), `data` (both as one data frame), `theta` and `phi` (the true input- and output-oriented Farrell efficiencies), `u`, `frontier` and `design`.

`dea_rate` returns a single negative number.

References

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

Park, B. U., Simar, L. and Weiner, C. (2000). The FDH estimator for productivity efficiency scores. *Econometric Theory* 16, 855–877.

Simar, L. and Wilson, P. W. (2015). Statistical approaches for non-parametric frontier models: a guided tour. *International Statistical Review* 83, 77–110.

See Also

`dea`, `dea_boot`

Examples

```
sim <- dea_sim(200, p = 1, q = 1, returns = 1, seed = 1)
sim

## A constant-returns fit recovers the truth on a constant-returns design.
fit <- dea(sim$x, sim$y, rts = "crs", orientation = "in", slack = FALSE)
c(bias = mean(fit$eff - sim$theta), rmse = sqrt(mean((fit$eff - sim$theta)^2)))

## What the rate ought to be, here and in a bigger problem.
dea_rate(1, 1, "crs")
dea_rate(4, 4, "vrs")
```

Index