Package 'hyperbolicDEA'

March 18, 2024

Type Package

Title Hyperbolic DEA Estimation

Version 1.0.0

Maintainer Alexander Öttl <alexoettl34@gmail.com>

Description Implements Data Envelopment Analysis (DEA) with a hyperbolic orientation using a non-linear programming solver. It enables flexible estimations with weight restrictions, non-discretionary variables, and a generalized distance function. Additionally, it allows for the calculation of slacks and super-efficiency scores. The methods are detailed in Öttl et al. (2023), <doi:10.1016/j.dajour.2023.100343>. Furthermore, the package provides a non-linear profitability estimation built upon the DEA framework.

License MIT + file LICENSE

Imports doParallel, dplyr, foreach, lpSolveAPI, nloptr, Benchmarking

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Alexander Öttl [cre, aut] (<https://orcid.org/0000-0002-0734-4135>), Daniel Gulde [aut]

Repository CRAN

Date/Publication 2024-03-18 17:30:07 UTC

R topics documented:

costDEA *Cost DEA model*

Description

Cost DEA model optimizing the input allocation with given prices. It returns the estimated lambdas as well as the optimal values for inputs and a cost efficiency score that is the ratio of optimal costs over observed costs.

Usage

```
costDEA(X, Y, pX, RTS = "crs")
```

Arguments

X	Vector, matrix or dataframe with DMUs as rows and inputs as columns
Y	Vector, matrix or dataframe with DMUs as rows and outputs as columns
pX	Vector, matrix or dataframe with prices for each DMU and input. Therefore it must have the same dimensions as X.
RTS	Character string indicating the returns-to-scale, e.g. "crs", "vrs".

Value

A list object containing the following:

lambdas	Estimated values for the composition of the respective Benchmarks. The lambdas are stored in a matrix with dimensions nrow(X) x nrow(X), where the row is the DMU under observation and the columns are the peers used for the Benchmark.
opt_value	Optimal inputs.
cost_eff	Cost efficiency as the ratio of the optimal cost to the observed cost.

See Also

[Benchmarking::cost.opt] for a similar function

Examples

```
X <- matrix(c(1,2,3,3,2,1,2,2), ncol = 2)
Y <- matrix(c(1,1,1,1), ncol = 1)

pX <- matrix(c(2,1,2,1,2,1,1,2), ncol =  2, byrow = TRUE)


cost_eff_input <- costDEA(X,Y,pX)
```

hyperbolicDEA *Hyperbolic estimation of DEA efficiency scores*

Description

Hyperbolic DEA implementation including weight restrictions, non-discretionary variables, gerenralized distance function, external referencing, estimation of slacks and super-efficiency scores. The mathematical and theoretical foundations of the code are presented in the paper "Data Envelopment Analysis and Hyperbolic Efficiency Measures: Extending Applications and Possiblities for Between-Group Comparisons" (2023) by Alexander Öttl, Mette Asmild, and Daniel Gulde.

Usage

```
hyperbolicDEA(
  X,
  Y,
  RTS = "vrs",
  WR = NULL,
  SLACK = FALSE,
  ACCURACY = 1e-10,
  XREF = NULL,
  YREF = NULL,
  SUPEREFF = FALSE,
  NONDISC_IN = NULL,
  NONDISC_OUT = NULL,
  PARALLEL = 1,
  ALPHA = 0.5
)
```

Arguments

X	Vector, matrix or dataframe with DMUs as rows and inputs as columns
Y	Vecotr, matrix or dataframe with DMUs as rows and outputs as columns
RTS	Character string indicating the returns-to-scale, e.g. "crs", "vrs", "ndrs", "nirs", "fdh"
WR	Matrix with one row per homogeneous linear weight restriction in standard form. The columns are ncol(WR) = ncol(Y) + ncol(X). Hence the first ncol(Y) columns are the restrictions on outputs and the last ncol(X) columns are the restrictions on inputs.
SLACK	Boolean variable indicating whether an additional estimation of slacks shall be performed when set to 'TRUE'. Be aware that SLACK estimation can change the lambda values.
ACCURACY	Accuracy value for non-linear programm solver.
XREF	Vector, matrix or dataframe with firms defining the technology as rows and inputs as columns

YREF	Vector, matrix or dataframe with firms defining the technology as rows and outputs as columns
SUPEREFF	Boolean variable indicating whether super-efficiencies shall be estimated
NONDISC_IN	Vector containing column indices of the input matrix that are non-discretionary variables e.g. c(1,3) so the first and the third input are non-discretionary
NONDISC_OUT	Vector containing column indices of the output matrix that are non-discretionary variables e.g. c(1,3) so the first and the third output are non-discretionary
PARALLEL	Integer of amount of cores that should be used for parallel computing (Check availability of computer)
ALPHA	ALPHA can be chosen between [0,1]. It indicates the relative weights given to the distance function to both outputs and inputs when approaching the frontier. More weight on the input orientation is set by alpha < 0.5. Here, the input efficiency score is estimated in the package. To receive the corresponding output efficiency score, estimate: e^((1-alpha)/alpha). Vice versa for an output weighted model alpha > 0.5. The output efficiency is given and the input efficiency can be recovered with: e^(alpha/(1-alpha))

Value

A list object containing the following information:

eff	Are the estimated efficiency scores for the DMUs under observation stored in a vector with the length nrow(X).
lambdas	Estimated values for the composition of the respective Benchmarks. The lambdas are stored in a matrix with the dimensions nrow(X) x nrow(X), where the row is the DMU under observation and the columns the peers used for the Benchmark.
mus	If WR != NULL, the estimated decision variables for the imposed weight restrictions are stored in a matrix with the dimensions nrow(X) x nrow(WR), where the rows are the DMUs and columns the weight restrictions. If the values are positive, the WR is binding for the respective DMU.
slack	If SLACK = TRUE, the slacks are estimated and stored in a matrix with the dimensions nrow(X) x (ncol(X) + ncol(Y)). Showing the Slack of each DMU (row) for each input and output (column).

Examples

```
X <- c(1,1,2,4,1.5,2,4,3)
Y <- c(1,2,4,4,0.5,2.5,3.5,4)
# we now impose linked weght restrictions. We assume outputs decrease by
# four units when inputs are reduced by one. And we assume that outputs can
# can be increased by one when inputs are increased by four

WR <- matrix(c(-4,-1,1,4), nrow = 2, byrow = TRUE)
hyperbolicDEA(X,Y,RTS="vrs", WR = WR)

# Another example having the same data but just estimate the results for DMU 1
# using XREF YREF and and a higher focus on inputs adjusting the ALPHA towards 0.
```

```
# Additionally, slacks are estimated.

hyperbolicDEA(X[1],Y[1],RTS="vrs", XREF = X, YREF = Y, WR = WR, ALPHA = 0.1, SLACK = TRUE)
```

`lprofitDEA` *Linear profit DEA model*

Description

Linear profit DEA model optimizing the difference of cost to revenue. It returns the estimated lambdas as well as the optimal values for inputs and outputs. The linear profit estimation does not account for scale differences and also has issues with negative profits. Therefore, it is recommended to use the non-linear profit model.

Usage

```
lprofitDEA(X, Y, pX, pY, RTS = "crs")
```

Arguments

`X`	Vector, matrix or dataframe with DMUs as rows and inputs as columns
`Y`	Vector, matrix or dataframe with DMUs as rows and outputs as columns
`pX`	Vector, matrix or dataframe with prices for each DMU and input. Therefore it must have the same dimensions as X.
`pY`	Vector, matrix or dataframe with prices for each DMU and output. Therefore it must have the same dimensions as Y.
`RTS`	Character string indicating the returns-to-scale, e.g. "crs", "vrs".

Value

A list object containing the following:

`lambdas`	Estimated values for the composition of the respective Benchmarks. The lambdas are stored in a matrix with dimensions nrow(X) x nrow(X), where the row is the DMU under observation and the columns are the peers used for the Benchmark.
`opt_value`	Optimal inputs and outputs.
`profit_eff`	Profit efficiency as the ratio of the differences between the observed and optimal difference of cost to revenue.

See Also

[Benchmarking::profit.opt] for a similar function

Examples

```
X <- matrix(c(1,2,3,3,2,1,2,2), ncol = 2)
Y <- matrix(c(1,1,1,1), ncol = 1)

pX <- matrix(c(2,1,2,1,2,1,1,2), ncol =  2, byrow = TRUE)
pY <- matrix(c(1,1,1,1), ncol = 1)

max_prof_lin<- lprofitDEA(X,Y,pX,pY)
```

`nlprofitDEA` *Non-linear profit DEA model*

Description

This function implements a non-linear profit DEA model that optimizes the ratio of cost over revenue given the prices for a DMU. It returns the estimated lambdas, optimal values for inputs and outputs, and a profit efficiency score. The profit efficiency score is calculated as the square root of the ratio of the observed revenue-cost ratio to the optimal revenue-cost ratio.

Usage

```
nlprofitDEA(X, Y, pX, pY, RTS = "crs")
```

Arguments

`X`	Vector, matrix or dataframe with DMUs as rows and inputs as columns.
`Y`	Vector, matrix or dataframe with DMUs as rows and outputs as columns.
`pX`	Vector, matrix or dataframe with prices for each DMU and input. It must have the same dimensions as X.
`pY`	Vector, matrix or dataframe with prices for each DMU and output. It must have the same dimensions as Y.
`RTS`	Character string indicating the returns-to-scale, e.g. "crs", "vrs".

Value

A list object containing the following:

`lambdas`	Estimated values for the composition of the respective Benchmarks. The lambdas are stored in a matrix with dimensions nrow(X) x nrow(X), where the row is the DMU under observation and the columns are the peers used for the Benchmark.
`opt_value`	Optimal inputs and outputs.
`profit_eff`	New profit efficiency score that accounts for simultaneous adjustments in inputs and outputs.

See Also

'deaprofitability()' function in the Julia package BenchmarkingEconomicEfficiency.jl.

Examples

```
X <- matrix(c(1,2,3,3,2,1,2,2), ncol = 2)
Y <- matrix(c(1,1,1,1), ncol = 1)

pX <- matrix(c(2,1,2,1,2,1,1,2), ncol =  2, byrow = TRUE)
pY <- matrix(c(1,1,1,1), ncol = 1)

max_prof_nolin <- nlprofitDEA(X,Y,pX,pY)
```

---

wrDEA *Estimation of DEA efficiency scores with linear input or output orientation and trade-off weight restrictions*

Description

Linear DEA estimation including the possibility of trade-off weight restrictions, external referencing, and super-efficiency scores. Furthermore, in a second stage slacks can be estimated. The function returns efficiency scores and adjusted lambdas according to the imposed weight restrictions and slack estimation. Additionally, mus are returned if weight restrictions are imposed that highlight binding restrictions for DMUs and the absolute slack values if slack-based estimation is applied.

Usage

```
wrDEA(
  X,
  Y,
  ORIENTATION = "out",
  RTS = "vrs",
  WR = NULL,
  XREF = NULL,
  YREF = NULL,
  SUPEREFF = FALSE,
  SLACK = FALSE
)
```

Arguments

X	Vector, matrix or dataframe with DMUs as rows and inputs as columns
Y	Vector, matrix or dataframe with DMUs as rows and outputs as columns
ORIENTATION	Character string indicating the orientation of the DEA model, e.g. "in", "out"

RTS	Character string indicating the returns-to-scale, e.g. "crs", "vrs", "ndrs", "nirs", "fdh"
WR	Matrix with one row per homogeneous linear weight restriction in standard form. The columns are ncol(WR) = ncol(Y) + ncol(X). Hence the first ncol(Y) columns are the restrictions on outputs and the last ncol(X) columns are the restrictions on inputs.
XREF	Vector, matrix or dataframe with firms defining the technology as rows and inputs as columns
YREF	Vector, matrix or dataframe with firms defining the technology as rows and outputs as columns
SUPEREFF	Boolean variable indicating whether super-efficiencies shall be estimated
SLACK	Boolean variable indicating whether slack-based estimation should be applied

Value

A list object containing the following information:

eff	Are the estimated efficiency scores for the DMUs under observation stored in a vector with the length nrow(X).
lambdas	Estimated values for the composition of the respective Benchmarks. The lambdas are stored in a matrix with the dimensions nrow(X) x nrow(X), where the row is the DMU under observation and the columns the peers used for the Benchmark. NOTE: Lambdas are automatically slack optimized.
mus	If WR != NULL, the estimated decision variables for the imposed weight restrictions are stored in a matrix with the dimensions nrow(X) x nrow(WR), where the rows are the DMUs and columns the weight restrictions. If the values are positive, the WR is binding for the respective DMU.
slack	If SLACK = TRUE, the slacks are estimated and stored in a matrix with the dimensions nrow(X) x (ncol(X) + ncol(Y)). Showing the Slack of each DMU (row) for each input and output (column).

Examples

```
X <- c(1,1,2,4,1.5,2,4,3)
Y <- c(1,2,4,4,0.5,2.5,3.5,4)

# Two weight restrictions in standard form first on output then input.
# The first WR shows the trade-off that inputs can be reduced by one unit
# which reduces outputs by four units. The second WR shows that outputs can
# be increased by one unit when inputs are increased by four units.

WR <- matrix(c(-4,-1,1,4), nrow = 2, byrow = TRUE)

wrDEA(X, Y, ORIENTATION = "in", RTS="vrs", WR = WR)

# For an estimation just focusing on one DMU one can for example use
# XREF and YREF to define the technology and then estimate the efficiency for
# the DMU under observation (here DMU 1). Let's additionally estimate the slacks.
```

```
wrDEA(X[1], Y[1], ORIENTATION = "in", RTS="vrs", XREF = X, YREF = Y, SLACK = TRUE, WR = WR)
```

Index