# Package: SdeaR (via r-universe)

October 29, 2026

**Type** Package

**Title** Stochastic Data Envelopment Analysis

**Version** 1.1.1

**Maintainer** Vicente Bolós <vicente.bolos@uv.es>

**Description** Set of functions for Stochastic Data Envelopment Analysis. Chance constrained versions of radial, directional and additive DEA models are implemented, as long as super-efficiency models. See: Cooper, W.W.; Deng, H.; Huang, Z.; Li, S.X. (2002). <doi:10.1057/palgrave.jors.2601433>, Bolós, V.J.; Benítez, R.; Coll-Serrano, V. (2024) <doi:10.1016/j.orp.2024.100307>.

**License** GPL

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.3

**Depends** R (>= 3.5)

**Imports** optiSolve, deaR, foreach

**Suggests** doParallel

**URL** https://github.com/vjbolos/SdeaR

**BugReports** https://github.com/vjbolos/SdeaR/issues

**NeedsCompilation** no

**Author** Vicente Bolós [aut, cre], Vicente Coll-Serrano [aut], Rafael Benítez Suárez [aut]

**Config/pak/sysreqs** cmake libfreetype6-dev libglpk-dev libglu1-mesa-dev make texlive libicu-dev libpng-dev libuv1-dev libxml2-dev libgl1-mesa-dev libssl-dev zlib1g-dev

**Repository** https://vjbolos.r-universe.dev

**Date/Publication** 2026-07-01 18:48:58 UTC

**RemoteUrl** https://github.com/vjbolos/sdear

**RemoteRef** HEAD

**RemoteSha** 393e08de3041bcadf16a4ddc94667d7f4adc8da7

# Contents

Automobile . . . . . 2
efficiencies.dea_stoch . . . . . 3
make_deadata_stoch . . . . . 4
modelstoch_additive . . . . . 9
modelstoch_additive_p . . . . . 13
modelstoch_addsupereff . . . . . 16
modelstoch_dir . . . . . 19
modelstoch_dir_dd . . . . . 23
modelstoch_radial . . . . . 27
modelstoch_radial_supereff . . . . . 32
Textile . . . . . 34

**Index** **36**

---

`Automobile` *Data: Cooper et al. (2001).*

---

## Description

Data of automobile industries collected from the Statistical Year Book of China of the Chinese Bureau of Statistics. Each year is treated as a DMU.

## Usage

```
data("Automobile")
```

## Format

Data frame with 17 rows and 4 columns. Definition of inputs (X) and output (Y):

**X1 = Labor** Expressed in units of 1000 persons.

**X2 = Capital** Stated in units of 1 million Ren Min Bi (Chinese monetary unit) adjusted to 1991 prices.

**Y = Output** Stated in units of 1 million Ren Min Bi (Chinese monetary unit) adjusted to 1991 prices.

## Author(s)

**Vicente Bolos** (<vicente.bolos@uv.es>). *Department of Business Mathematics*

**Rafael Benitez** (<rafael.suarez@uv.es>). *Department of Business Mathematics*

**Vicente Coll-Serrano** (<vicente.coll@uv.es>). *Quantitative Methods for Measuring Culture (MC2). Applied Economics.*

University of Valencia (Spain)

## Source

Cooper, W.W.; Denga, H.; Gua, B.; Lib, S.; Thrall, R.M. (2001). "Using DEA to improve the management of congestion in Chinese industries (1981-1997)", Socio-Economic Planning Sciences, 35(4), 227-242.

## See Also

make_deadata_stoch, modelstoch_radial

---

# efficiencies.dea_stoch

*Objective scores*

---

## Description

Extract the scores (optimal objective values) of the evaluated DMUs from a stochastic DEA solution. Note that these scores may not always be interpreted as efficiencies.

## Usage

```
## S3 method for class 'dea_stoch'
efficiencies(x, ...)
```

## Arguments

| | |
|---|---|
| x | Object of class dea_stoch obtained with some of the stochastic DEA modelstoch_* functions. |
| ... | Other options (for compatibility reasons) |

## Value

A matrix with the scores (optimal objective values).

## Author(s)

**Vicente Bolós** (<vicente.bolos@uv.es>). *Department of Business Mathematics*

**Rafael Benítez** (<rafael.suarez@uv.es>). *Department of Business Mathematics*

**Vicente Coll-Serrano** (<vicente.coll@uv.es>). *Quantitative Methods for Measuring Culture (MC2). Applied Economics.*

University of Valencia (Spain)

## Examples

```
# Example 1.
library(deaR)
data("Coll_Blasco_2006")
ni <- 2 # number of inputs
no <- 2 # number of outputs
data_example <- make_deadata(datadea = Coll_Blasco_2006,
                             ni = ni,
                             no = no)
nd <- length(data_example$dmunames) # number of DMUs
var_input <- matrix(1, nrow = ni, ncol = nd)
var_output <- matrix(1, nrow = no, ncol = nd)
data_stoch <- make_deadata_stoch(datadea = data_example,
                                 var_input = var_input,
                                 var_output = var_output)
# Evaluate the sixth DMU
Collstoch <- modelstoch_radial(data_stoch, dmu_eval = 6)
efficiencies(Collstoch)
```

---

`make_deadata_stoch` *make_deadata_stoch*

---

## Description

This function creates, from a `deadata` object, a `deadata_stoch` object by adding the corresponding covariance matrices. These objects are prepared to be passed to a modelstoch_* function.

We consider $\mathcal{D} = \{\text{DMU}_1, \ldots, \text{DMU}_n\}$ a set of $n$ DMUs with $m$ stochastic inputs and $s$ stochastic outputs. Matrices $\tilde{X} = (\tilde{x}_{ij})$ and $\tilde{Y} = (\tilde{y}_{rj})$ are the input and output data matrices, respectively, where $\tilde{x}_{ij}$ and $\tilde{y}_{rj}$ represent the $i$-th input and $r$-th output of the $j$-th DMU. Moreover, we denote by $X = (x_{ij})$ and $Y = (y_{rj})$ their expected values. We suppose that $\tilde{x}_{ij}$ and $\tilde{y}_{rj}$ follow a multivariate probability distribution with means $E(\tilde{x}_{ij}) = x_{ij}$, $E(\tilde{y}_{rj}) = y_{rj}$ and covariance matrix

$$\Delta = \begin{pmatrix} \Delta_{11}^{II} & \cdots & \Delta_{1m}^{II} & \Delta_{11}^{IO} & \cdots & \Delta_{1s}^{IO} \\ \vdots & & \vdots & \vdots & & \vdots \\ \Delta_{m1}^{II} & \cdots & \Delta_{mm}^{II} & \Delta_{m1}^{IO} & \cdots & \Delta_{ms}^{IO} \\ \Delta_{11}^{OI} & \cdots & \Delta_{1m}^{OI} & \Delta_{11}^{OO} & \cdots & \Delta_{1s}^{OO} \\ \vdots & & \vdots & \vdots & & \vdots \\ \Delta_{s1}^{OI} & \cdots & \Delta_{sm}^{OI} & \Delta_{s1}^{OO} & \cdots & \Delta_{ss}^{OO} \end{pmatrix}_{n(m+s)\times n(m+s)}$$

where $\Delta_{ik}^{II}$, $\Delta_{rp}^{OO}$, $\Delta_{ir}^{IO}$ and $\Delta_{ri}^{OI}$ are $n \times n$ matrices, for $1 \leq i, k \leq m$ and $1 \leq r, p \leq s$, such that

$$\left(\Delta_{ik}^{II}\right)_{jq} = \text{Cov}(\tilde{x}_{ij}, \tilde{x}_{kq}),$$

$$\left(\Delta_{rp}^{OO}\right)_{jq} = \text{Cov}(\tilde{y}_{rj}, \tilde{y}_{pq}),$$

$$\left(\Delta_{ir}^{IO}\right)_{jq} = \left(\Delta_{ri}^{OI}\right)_{qj} = \mathrm{Cov}(\tilde{x}_{ij}, \tilde{y}_{rq}),$$

with $1 \leq j, q \leq n$.

- If we have the covariances matrix in the general form above, it can be introduced directly by parameter `cov_matrix`. Since this matrix is supposed to be symmetric, only values above the diagonal are read, ignoring values below the diagonal.

- Alternatively, we can introduce the covariances matrix using parameters `cov_II`, `cov_OO` and `cov_IO`, that are 4-dimensional arrays of size $m \times m \times n \times n$, $s \times s \times n \times n$ and $m \times s \times n \times n$, respectively, such that

$$\texttt{cov_II[i, k, , ]} = \Delta_{ik}^{II},$$

$$\texttt{cov_OO[r, p, , ]} = \Delta_{rp}^{OO},$$

$$\texttt{cov_IO[i, r, , ]} = \Delta_{ir}^{IO},$$

for $1 \leq i, k \leq m$ and $1 \leq r, p \leq s$. Since matrices $\Delta_{ii}^{II}$ and $\Delta_{rr}^{OO}$ are supposed to be symmetric, only values above the diagonal are read, ignoring values below the diagonal. Moreover, since $\Delta_{ki}^{II}$ is the transpose of $\Delta_{ik}^{II}$, and $\Delta_{pr}^{OO}$ is the transpose of $\Delta_{rp}^{OO}$, only matrices $\Delta_{ik}^{II}$ and $\Delta_{rp}^{OO}$ with $i \leq k$ and $r \leq p$ are necessary, ignoring those with $i > k$ and $r > p$.

- If covariances between different inputs/outputs are zero, we can make use of parameters `cov_input` and `cov_output`, that are arrays of size $m \times n \times n$ and $s \times n \times n$, respectively, such that

$$\texttt{cov_input[i, , ]} = \Delta_{ii}^{II},$$

$$\texttt{cov_output[r, , ]} = \Delta_{rr}^{OO},$$

for $1 \leq i \leq m$ and $1 \leq r \leq s$. By symmetry of $\Delta_{ii}^{II}$ and $\Delta_{rr}^{OO}$, only values above the diagonal are read, ignoring values below the diagonal.

- Finally, if all the variables are independent then the covariances matrix is diagonal. Hence, we might use parameters `var_input` and `var_output`, that are matrices of size $m \times n$ and $s \times n$, respectively, such that

$$\texttt{var_input[i, j]} = \mathrm{Var}\left(\tilde{x}_{ij}\right),$$

$$\texttt{var_output[r, j]} = \mathrm{Var}\left(\tilde{y}_{rj}\right),$$

for $1 \leq i \leq m$, $1 \leq r \leq s$ and $1 \leq j \leq n$.

## Usage

```
make_deadata_stoch(datadea = NULL,
                   var_input = NULL,
                   var_output = NULL,
                   cov_input = NULL,
                   cov_output = NULL,
                   cov_II = NULL,
                   cov_OO = NULL,
                   cov_IO = NULL,
                   cov_matrix = NULL)
```

## Arguments

| | |
|---|---|
| datadea | The deadata object. |
| var_input | A matrix of size m x n, where m is the number of inputs and n is the number of DMUs. It contains the variances of each input of each DMU, such that var_input[i, j] is the variance of the input i of DMU j. Use this parameter if all covariances are 0. |
| var_output | A matrix of size s x n, where s is the number of outputs and n is the number of DMUs. It contains the variances of each output of each DMU, such that var_output[r, j] is the variance of the output r of DMU j. Use this parameter if all covariances are 0. |
| cov_input | An array of size m x n x n containing the covariances of each input between DMUs, such that cov_input[i, j, k] is the covariance between the input i of DMU j and the input i of DMU k. The corresponding variances are in the diagonal of each n x n submatrix. Since these submatrices are supposed to be symmetric, only values above the diagonal are read in order to reconstruct the symmetric submatrices, ignoring values below the diagonal. Use this parameter if covariances between different inputs are 0. |
| cov_output | An array of size s x n x n containing the covariances of each output between DMUs, such that cov_output[r, j, k] is the covariance between the output r of DMU j and the output r of DMU k. The corresponding variances are in the diagonal of each n x n submatrix. Since these submatrices are supposed to be symmetric, only values above the diagonal are read in order to reconstruct the symmetric submatrices, ignoring values below the diagonal. Use this parameter if covariances between different outputs are 0. |
| cov_II | An array of size m x m x n x n containing the covariances between inputs and between DMUs, such that cov_II[i1, i2, j, k] is the covariance between the input i1 of DMU j and the input i2 of DMU k. For the input i, the corresponding variances are in the diagonal of the n x n submatrices of the form cov_II[i, i, , ]. Since these submatrices are supposed to be symmetric, only values above the diagonal are read in order to reconstruct the symmetric submatrices, ignoring values below the diagonal. Moreover, since cov_II[i2, i1, , ] is the transpose of cov_II[i1, i2, , ], only submatrices cov_II[i1, i2, , ] with i1 <= i2 are necessary, ignoring those with i1 > i2. Use this parameter if covariances between inputs are nonzero. |
| cov_OO | An array of size s x s x n x n containing the covariances between outputs and between DMUs, such that cov_OO[r1, r2, j, k] is the covariance between the output r1 of DMU j and the output r2 of DMU k. For the output r, the corresponding variances are in the diagonal of the n x n submatrices of the form cov_OO[r, r, , ]. Since these submatrices are supposed to be symmetric, only values above the diagonal are read in order to reconstruct the symmetric submatrices, ignoring values below the diagonal. Moreover, since cov_OO[r2, r1, , ] is the transpose of cov_OO[r1, r2, , ], only submatrices cov_OO[r1, r2, , ] with r1 <= r2 are necessary, ignoring those with r1 > r2. Use this parameter if covariances between outputs are nonzero. |
| cov_IO | An array of size m x s x n x n containing the covariances between inputs and outputs, and between DMUs, such that cov_IO[i, r, j, k] is the covariance |

between the input i of DMU j and the output r of DMU k. Use this parameter if covariances between inputs and outputs are nonzero.

cov_matrix A matrix of size n(m+s) x n(m+s) following the notation of Cooper et al. (1998). Since this matrix is supposed to be symmetric, only values above the diagonal are read, ignoring values below the diagonal.

## Value

An object of class deadata_stoch.

## Author(s)

**Vicente Bolós** (<vicente.bolos@uv.es>). *Department of Business Mathematics*

**Rafael Benítez** (<rafael.suarez@uv.es>). *Department of Business Mathematics*

**Vicente Coll-Serrano** (<vicente.coll@uv.es>). *Quantitative Methods for Measuring Culture (MC2). Applied Economics.*

University of Valencia (Spain)

## References

Cooper, W.W.; Huang, Z.; Lelas, V.; Li, S.X.; Olesen, O.B. (1998). "Chance Constrained Programming Formulations for Stochastic Characterizations of Efficiency and Dominance in DEA", Journal of Productivity Analysis, 9, 53-79.

Land, K.C; Lovell, C.A.K.; Thore, S. (1993). "Chance-constrained data envelopment analysis", Managerial and Decision Economics, Vol. 14, No. 6, 541-554.

## Examples

```
# Example 1.

library(deaR)
data("Coll_Blasco_2006")
ni <- 2 # number of inputs
no <- 2 # number of outputs
data_example <- make_deadata(datadea = Coll_Blasco_2006,
                             ni = ni,
                             no = no)
nd <- length(data_example$dmunames) # number of DMUs
# All variances are 1.
var_input <- matrix(1, nrow = ni, ncol = nd)
var_output <- matrix(1, nrow = no, ncol = nd)
data_stoch <- make_deadata_stoch(datadea = data_example,
                                 var_input = var_input,
                                 var_output = var_output)
# All covariances are 1.
cov_input <- array(1, dim = c(ni, nd, nd))
cov_output <- array(1, dim = c(no, nd, nd))
data_stoch2 <- make_deadata_stoch(datadea = data_example,
                                  cov_input = cov_input,
                                  cov_output = cov_output)
```

```
# Example 2. Deterministic data with one stochastic input.

library(deaR)
dmunames <- c("A", "B", "C")
nd <- length(dmunames) # Number of DMUs
inputnames <- c("Input_1", "Input_2")
ni <- length(inputnames) # Number of Inputs
outputnames <- c("Output_1", "Output_2", "Output_3")
no <- length(outputnames) # Number of Outputs
X <- matrix(c(5, 14, 8, 15, 7, 12),
            nrow = ni, ncol = nd, dimnames = list(inputnames, dmunames))
Y <- matrix(c(9, 4, 16, 5, 7, 10, 4, 9, 13),
            nrow = no, ncol = nd, dimnames = list(outputnames, dmunames))
datadea <- make_deadata(inputs = X,
                        outputs = Y)
covX <- array(0, dim = c(2, 3, 3))
# The 2nd input is stochastic.
# Since the corresponding 3x3 covariances matrix is symmetric, only values
# above the diagonal are necessary.
covX[2, 1, ] <- c(1.4, 0.9, 0.6)
covX[2, 2, 2:3] <- c(1.5, 0.7)
covX[2, 3, 3] <- 1.2
# Alternatively (note that values below the diagonal are ignored).
covX[2, , ] <- matrix(c(1.4, 0.9, 0.6, 0, 1.5, 0.7, 0, 0, 1.2),
                      nrow = 3,
                      byrow = TRUE)
datadea_stoch <- make_deadata_stoch(datadea,
                                    cov_input = covX)

# Example 3. Replication of Program Follow Through data in Land et al. (1993).

library(deaR)
data("PFT1981")
# Selecting DMUs in Program Follow Through (PFT)
PFT <- PFT1981[1:49, ]
PFT <- make_deadata(PFT,
                    inputs = 2:6,
                    outputs = 7:9)
c <- 0.5
var_output <- matrix(c^2, nrow = 3, ncol = 49)
PFT_stoch <- make_deadata_stoch(datadea = PFT, var_output = var_output)

# Example 4. Random observations.

library(deaR)

nobs <- 5 # number of observations
ni <- 2   # number of inputs
no <- 2   # number of outputs
nd <- 3   # number of DMUs

# Generator function
```

```
gen_df <- function(nobs, nd) {
  as.data.frame(
    replicate(nd, rnorm(nobs, mean = sample(5:10, 1)))
  )
}

# Generate random observations
inputs_list <- lapply(seq_len(ni), \(i) {
  df <- gen_df(nobs, nd)
  names(df) <- paste0("I", i, "D", seq_len(nd))
  df
})
outputs_list <- lapply(seq_len(no), \(i) {
  df <- gen_df(nobs, nd)
  names(df) <- paste0("O", i, "D", seq_len(nd))
  df
})

# Generate deadata with means of observations
inputs <- do.call(
  rbind,
  lapply(inputs_list, colMeans)
)
rownames(inputs)  <- paste0("I", seq_len(ni))
colnames(inputs)  <- paste0("D", seq_len(nd))
outputs <- do.call(
  rbind,
  lapply(outputs_list, colMeans)
)
rownames(outputs) <- paste0("O", seq_len(no))
colnames(outputs) <- colnames(inputs)
datadea <- make_deadata(inputs = inputs,
                        outputs = outputs)

# Generate covariances matrix
cov_matrix <- cov(
  do.call(
    cbind,
    c(inputs_list, outputs_list)
  )
)

# Generate deadata_stoch
datadea_stoch <- make_deadata_stoch(datadea,
                                    cov_matrix = cov_matrix)
```

---

`modelstoch_additive` *Chance Constrained Additive E-model.*

---

## Description

It solves the chance constrained additive E-model based on the deterministic additive model from Charnes et al. (1985), under constant and non-constant returns to scale.

Besides, the user can set weights for the input and/or output slacks. So, it is also possible to solve chance constrained versions of weighted additive models like Measure of Inefficiency Proportions (MIP) or Range Adjusted Measure (RAM), see Cooper et al. (1999).

We consider $\mathcal{D} = \{\text{DMU}_1, \ldots, \text{DMU}_n\}$ a set of $n$ DMUs with $m$ stochastic inputs and $s$ stochastic outputs. Matrices $\tilde{X} = (\tilde{x}_{ij})$ and $\tilde{Y} = (\tilde{y}_{rj})$ are the input and output data matrices, respectively, where $\tilde{x}_{ij}$ and $\tilde{y}_{rj}$ represent the $i$-th input and $r$-th output of the $j$-th DMU. Moreover, we denote by $X = (x_{ij})$ and $Y = (y_{rj})$ their expected values. In general, we denote vectors by bold-face letters and they are considered as column vectors unless otherwise stated. The 0-vector is denoted by $\mathbf{0}$ and the context determines its dimension.

Given $0 < \alpha < 1$, the program for $\text{DMU}_o$ with constant returns to scale is given by

$$\begin{aligned} \max_{\boldsymbol{\lambda},\mathbf{s}^-,\mathbf{s}^+} \quad & \mathbf{w}^-\mathbf{s}^- + \mathbf{w}^+\mathbf{s}^+ \\ \text{s.t.} \quad & P\left\{\left(\tilde{\mathbf{x}}_o - \tilde{X}\boldsymbol{\lambda} - \mathbf{s}^-\right)_i \geq 0\right\} \geq 1-\alpha, \qquad i = 1,\ldots,m, \\ & P\left\{\left(\tilde{Y}\boldsymbol{\lambda} - \tilde{\mathbf{y}}_o - \mathbf{s}^+\right)_r \geq 0\right\} \geq 1-\alpha, \qquad r = 1,\ldots,s, \\ & \boldsymbol{\lambda} \geq \mathbf{0},\ \mathbf{s}^- \geq \mathbf{0},\ \mathbf{s}^+ \geq \mathbf{0}, \end{aligned}$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)^\top$, $\tilde{\mathbf{x}}_o = (\tilde{x}_{1o}, \ldots, \tilde{x}_{mo})^\top$ and $\tilde{\mathbf{y}}_o = (\tilde{y}_{1o}, \ldots, \tilde{y}_{so})^\top$ are column vectors. Moreover, $\mathbf{s}^-, \mathbf{s}^+$ are column vectors with the slacks, and $\mathbf{w}^-, \mathbf{w}^+$ are positive row vectors with the weights for the slacks. Different returns to scale can be easily considered by adding the corresponding constraints: $\mathbf{e}\boldsymbol{\lambda} = 1$ (VRS), $0 \leq \mathbf{e}\boldsymbol{\lambda} \leq 1$ (NIRS), $\mathbf{e}\boldsymbol{\lambda} \geq 1$ (NDRS) or $L \leq \mathbf{e}\boldsymbol{\lambda} \leq U$ (GRS), with $0 \leq L \leq 1$ and $U \geq 1$, where $\mathbf{e} = (1, \ldots, 1)$ is a row vector.

The deterministic equivalent for a multivariate normal distribution of inputs/outputs is given by

$$\begin{aligned} \max_{\boldsymbol{\lambda},\mathbf{s}^-,\mathbf{s}^+} \quad & \mathbf{w}^-\mathbf{s}^- + \mathbf{w}^+\mathbf{s}^+ \\ \text{s.t.} \quad & \mathbf{x}_o - X\boldsymbol{\lambda} - \mathbf{s}^- + \Phi^{-1}(\alpha)\boldsymbol{\sigma}^-(\boldsymbol{\lambda}) \geq \mathbf{0}, \\ & Y\boldsymbol{\lambda} - \mathbf{y}_o - \mathbf{s}^+ + \Phi^{-1}(\alpha)\boldsymbol{\sigma}^+(\boldsymbol{\lambda}) \geq \mathbf{0}, \\ & \boldsymbol{\lambda} \geq \mathbf{0},\ \mathbf{s}^- \geq \mathbf{0},\ \mathbf{s}^+ \geq \mathbf{0}, \end{aligned}$$

where $\Phi$ is the standard normal distribution, and

$$\begin{aligned} \left(\sigma_i^-(\boldsymbol{\lambda})\right)^2 &= \sum_{j,q=1}^{n} \lambda_j\lambda_q \text{Cov}(\tilde{x}_{ij}, \tilde{x}_{iq}) - 2\sum_{j=1}^{n} \lambda_j \text{Cov}(\tilde{x}_{ij}, \tilde{x}_{io}) \\ &\quad + \text{Var}(\tilde{x}_{io}), \quad i = 1,\ldots,m, \\ \left(\sigma_r^+(\boldsymbol{\lambda})\right)^2 &= \sum_{j,q=1}^{n} \lambda_j\lambda_q \text{Cov}(\tilde{y}_{rj}, \tilde{y}_{rq}) - 2\sum_{j=1}^{n} \lambda_j \text{Cov}(\tilde{y}_{rj}, \tilde{y}_{ro}) \\ &\quad + \text{Var}(\tilde{y}_{ro}), \quad r = 1,\ldots,s. \end{aligned}$$

## Usage

```
modelstoch_additive(datadea,
                    alpha = 0.05,
                    dmu_eval = NULL,
                    dmu_ref = NULL,
                    orientation = NULL,
                    weight_slack_i = 1,
                    weight_slack_o = 1,
                    rts = c("crs", "vrs", "nirs", "ndrs", "grs"),
                    L = 1,
                    U = 1,
                    solver = c("alabama"),
                    X = NULL,
                    give_X = FALSE,
                    n_attempts_max = 5,
                    returnqp = FALSE,
                    parallel = FALSE,
                    ...)
```

## Arguments

| | |
|---|---|
| datadea | The data of class deadata_stoch, including n DMUs, and the expected values of m inputs and s outputs. |
| alpha | A value for parameter alpha. |
| dmu_eval | A numeric vector containing which DMUs have to be evaluated. If NULL (default), all DMUs are considered. |
| dmu_ref | A numeric vector containing which DMUs are the evaluation reference set. If NULL (default), all DMUs are considered. |
| orientation | This parameter is either NULL (default) or a string, equal to "io" (input-oriented) or "oo" (output-oriented). It is used to modify the weight slacks. If input-oriented, weight_slack_o are taken 0. If output-oriented, weight_slack_i are taken 0. |
| weight_slack_i | A value, vector of length m, or matrix m x ne (where ne is the length of dmu_eval) with the weights of the input slacks. If 0, output-oriented. |
| weight_slack_o | A value, vector of length s, or matrix s x ne (where ne is the length of dmu_eval) with the weights of the output slacks. If 0, input-oriented. |
| rts | A string, determining the type of returns to scale, equal to "crs" (constant), "vrs" (variable), "nirs" (non-increasing), "ndrs" (non-decreasing) or "grs" (generalized). |
| L | Lower bound for the generalized returns to scale (grs). |
| U | Upper bound for the generalized returns to scale (grs). |
| solver | Character string with the name of the solver used by function solvecop from package optiSolve. Only alabama solver is currently operational, but more solvers could be implemented in the future. |

| | |
|---|---|
| X | Initial vector for the solver in the first attempt. The variables are lambdas of reference DMUs, input slacks, output slacks, input sigmas and output sigmas. It can be a matrix of ne rows (where ne is the length of dmu_eval), with different initial vectors for each evaluated DMU. If it is a vector, the same initial vector is applied to all evaluated DMUs. |
| give_X | Logical. If it is TRUE and X is NULL, it makes the initial vector to be the variables corresponding to each evaluated DMU. If it is FALSE (default) and X is NULL, the initial vector is given internally by the solver (usually randomly generated). |
| n_attempts_max | A value with the maximum number of attempts if the solver does not converge. Each attempt uses a different initial vector. |
| returnqp | Logical. If it is TRUE, it returns the quadratic problems (objective function and constraints). |
| parallel | Logical, if TRUE, the DMUs are computed in parallel (default FALSE). |
| ... | Other parameters to be passed to the solver. |

## Value

A list with the results for the evaluated DMUs and other parameters for reproducibility.

## Note

A DMU is $\alpha$-stochastically efficient if and only if the optimal objective value of the problem, (objval), is zero (or less than zero).

## Author(s)

**Vicente Bolós** (<vicente.bolos@uv.es>). *Department of Business Mathematics*

**Rafael Benítez** (<rafael.suarez@uv.es>). *Department of Business Mathematics*

**Vicente Coll-Serrano** (<vicente.coll@uv.es>). *Quantitative Methods for Measuring Culture (MC2). Applied Economics.*

University of Valencia (Spain)

## References

Charnes, A.; Cooper, W.W.; Golany, B.; Seiford, L.; Stuz, J. (1985) "Foundations of Data Envelopment Analysis for Pareto-Koopmans Efficient Empirical Production Functions", Journal of Econometrics, 30(1-2), 91-107. doi:10.1016/03044076(85)901332

Cooper, W.W.; Park, K.S.; Pastor, J.T. (1999). "RAM: A Range Adjusted Measure of Inefficiencies for Use with Additive Models, and Relations to Other Models and Measures in DEA". Journal of Productivity Analysis, 11, p. 5-42. doi:10.1023/A:1007701304281

`modelstoch_additive_p` *Chance Constrained Additive P-model.*

## Description

It solves the "max" version of the "almost 100 chance constrained problem from Cooper et al. (1998), under constant and non-constant returns to scale.

Besides, the user can set weights for the input and/or output slacks. So, it is also possible to solve chance constrained versions of weighted additive models like Measure of Inefficiency Proportions (MIP) or Range Adjusted Measure (RAM), see Cooper et al. (1999).

We consider $\mathcal{D} = \{\text{DMU}_1, \ldots, \text{DMU}_n\}$ a set of $n$ DMUs with $m$ stochastic inputs and $s$ stochastic outputs. Matrices $\tilde{X} = (\tilde{x}_{ij})$ and $\tilde{Y} = (\tilde{y}_{rj})$ are the input and output data matrices, respectively, where $\tilde{x}_{ij}$ and $\tilde{y}_{rj}$ represent the $i$-th input and $r$-th output of the $j$-th DMU. Moreover, we denote by $X = (x_{ij})$ and $Y = (y_{rj})$ their expected values. In general, we denote vectors by bold-face letters and they are considered as column vectors unless otherwise stated. The 0-vector is denoted by $\mathbf{0}$ and the context determines its dimension.

Given $0 < \alpha < 1$ and $\varepsilon$ a positive non-Archimedean infinitesimal, the program for $\text{DMU}_o$ with constant returns to scale is given by

$$\begin{aligned}
\max_{\boldsymbol{\lambda}} \quad & P\left\{\mathbf{w}^- \cdot \left(\tilde{\mathbf{x}}_o - \tilde{X}\boldsymbol{\lambda}\right) + \mathbf{w}^+ \cdot \left(\tilde{Y}\boldsymbol{\lambda} - \tilde{\mathbf{y}}_o\right) > 0\right\} \\
\text{s.t.} \quad & P\left\{\left(\tilde{\mathbf{x}}_o - \tilde{X}\boldsymbol{\lambda}\right)_i > 0\right\} \geq 1 - \varepsilon, \quad i = 1, \ldots, m, \\
& P\left\{\left(\tilde{Y}\boldsymbol{\lambda} - \tilde{\mathbf{y}}_o\right)_r > 0\right\} \geq 1 - \varepsilon, \quad r = 1, \ldots, s, \\
& \boldsymbol{\lambda} \geq \mathbf{0},
\end{aligned}$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)^\top$, $\tilde{\mathbf{x}}_o = (\tilde{x}_{1o}, \ldots, \tilde{x}_{mo})^\top$ and $\tilde{\mathbf{y}}_o = (\tilde{y}_{1o}, \ldots, \tilde{y}_{so})^\top$ are column vectors. Moreover, $\mathbf{w}^-, \mathbf{w}^+$ are positive row vectors with the weights for the slacks. Different returns to scale can be easily considered by adding the corresponding constraints: $\mathbf{e}\boldsymbol{\lambda} = 1$ (VRS), $0 \leq \mathbf{e}\boldsymbol{\lambda} \leq 1$ (NIRS), $\mathbf{e}\boldsymbol{\lambda} \geq 1$ (NDRS) or $L \leq \mathbf{e}\boldsymbol{\lambda} \leq U$ (GRS), with $0 \leq L \leq 1$ and $U \geq 1$, where $\mathbf{e} = (1, \ldots, 1)$ is a row vector.

The deterministic equivalent for a multivariate normal distribution of inputs/outputs is given by

$$\begin{aligned}
\max_{\boldsymbol{\lambda}} \quad & \mathbf{w}^- \cdot (\mathbf{x}_o - X\boldsymbol{\lambda}) + \mathbf{w}^+ \cdot (Y\boldsymbol{\lambda} - \mathbf{y}_o) - \Phi^{-1}(\alpha)\sigma(\boldsymbol{\lambda}) \\
\text{s.t.} \quad & \mathbf{x}_o - X\boldsymbol{\lambda} + \Phi^{-1}(\varepsilon)\boldsymbol{\sigma}^-(\boldsymbol{\lambda}) \geq \mathbf{0}, \\
& Y\boldsymbol{\lambda} - \mathbf{y}_o + \Phi^{-1}(\varepsilon)\boldsymbol{\sigma}^+(\boldsymbol{\lambda}) \geq \mathbf{0}, \\
& \boldsymbol{\lambda} \geq \mathbf{0},
\end{aligned}$$

where $\Phi$ is the standard normal distribution, and

$$\left(\sigma\left(\boldsymbol{\lambda}\right)\right)^2 = \boldsymbol{\lambda}^\top \left[\sum_{i,k=1}^{m} \Delta_{ik}^{II} + \sum_{r,p=1}^{s} \Delta_{rp}^{OO} - 2\sum_{i=1}^{m}\sum_{r=1}^{s} \Delta_{ir}^{IO}\right] \boldsymbol{\lambda}$$

$$+2\sum_{j=1}^{n}\lambda_j\left[\sum_{i=1}^{m}\sum_{r=1}^{s}\left(\mathrm{Cov}(\tilde{x}_{io},\tilde{y}_{rj})+\mathrm{Cov}(\tilde{x}_{ij},\tilde{y}_{ro})\right)\right.$$

$$\left.-\sum_{i,k=1}^{m}\mathrm{Cov}(\tilde{x}_{io},\tilde{x}_{kj})-\sum_{r,p=1}^{s}\mathrm{Cov}(\tilde{y}_{ro},\tilde{y}_{pj})\right]$$

$$+\sum_{i,k=1}^{m}\mathrm{Cov}(\tilde{x}_{io},\tilde{x}_{ko})+\sum_{r,p=1}^{s}\mathrm{Cov}(\tilde{y}_{ro},\tilde{y}_{po})-2\sum_{i=1}^{m}\sum_{r=1}^{s}\mathrm{Cov}(\tilde{x}_{io},\tilde{y}_{ro}),$$

$$\left(\sigma_i^{-}\left(\boldsymbol{\lambda}\right)\right)^2=\boldsymbol{\lambda}^{\top}\Delta_{ii}^{II}\boldsymbol{\lambda}-2\sum_{j=1}^{n}\lambda_j\mathrm{Cov}(\tilde{x}_{ij},\tilde{x}_{io})+\mathrm{Var}(\tilde{x}_{io}),\quad i=1,\ldots,m,$$

$$\left(\sigma_r^{+}\left(\boldsymbol{\lambda}\right)\right)^2=\boldsymbol{\lambda}^{\top}\Delta_{rr}^{OO}\boldsymbol{\lambda}-2\sum_{j=1}^{n}\lambda_j\mathrm{Cov}(\tilde{y}_{rj},\tilde{y}_{ro})+\mathrm{Var}(\tilde{y}_{ro}),\quad r=1,\ldots,s.$$

such that

$$\left(\Delta_{ik}^{II}\right)_{jq}=\mathrm{Cov}(\tilde{x}_{ij},\tilde{x}_{kq}),$$

$$\left(\Delta_{rp}^{OO}\right)_{jq}=\mathrm{Cov}(\tilde{y}_{rj},\tilde{y}_{pq}),$$

$$\left(\Delta_{ir}^{IO}\right)_{jq}=\left(\Delta_{ri}^{OI}\right)_{qj}=\mathrm{Cov}(\tilde{x}_{ij},\tilde{y}_{rq}),$$

with $1\leq j,q\leq n$.

## Usage

```
modelstoch_additive_p(datadea,
                      alpha = 0.05,
                      epsilon = 0.05,
                      dmu_eval = NULL,
                      dmu_ref = NULL,
                      orientation = NULL,
                      weight_slack_i = 1,
                      weight_slack_o = 1,
                      rts = c("crs", "vrs", "nirs", "ndrs", "grs"),
                      L = 1,
                      U = 1,
                      solver = c("alabama"),
                      X = NULL,
                      give_X = FALSE,
                      n_attempts_max = 5,
                      returnqp = FALSE,
                      parallel = FALSE,
                      ...)
```

## Arguments

| | |
|---|---|
| datadea | The data of class deadata_stoch, including n DMUs, and the expected values of m inputs and s outputs. |
| alpha | A value for parameter alpha. |
| epsilon | A value for parameter epsilon. |
| dmu_eval | A numeric vector containing which DMUs have to be evaluated. If NULL (default), all DMUs are considered. |
| dmu_ref | A numeric vector containing which DMUs are the evaluation reference set. If NULL (default), all DMUs are considered. |
| orientation | This parameter is either NULL (default) or a string, equal to "io" (input-oriented) or "oo" (output-oriented). It is used to modify the weight slacks. If input-oriented, weight_slack_o are taken 0. If output-oriented, weight_slack_i are taken 0. |
| weight_slack_i | A value, vector of length m, or matrix m x ne (where ne is the length of dmu_eval) with the weights of the input slacks. If 0, output-oriented. |
| weight_slack_o | A value, vector of length s, or matrix s x ne (where ne is the length of dmu_eval) with the weights of the output slacks. If 0, input-oriented. |
| rts | A string, determining the type of returns to scale, equal to "crs" (constant), "vrs" (variable), "nirs" (non-increasing), "ndrs" (non-decreasing) or "grs" (generalized). |
| L | Lower bound for the generalized returns to scale (grs). |
| U | Upper bound for the generalized returns to scale (grs). |
| solver | Character string with the name of the solver used by function solvecop from package optiSolve. Only alabama solver is currently operational, but more solvers could be implemented in the future. |
| X | Initial vector for the solver in the first attempt. The variables are lambdas of reference DMUs, input sigmas, output sigmas and sigma. It can be a matrix of ne rows (where ne is the length of dmu_eval), with different initial vectors for each evaluated DMU. If it is a vector, the same initial vector is applied to all evaluated DMUs. |
| give_X | Logical. If it is TRUE and X is NULL, it makes the initial vector to be the variables corresponding to each evaluated DMU. If it is FALSE (default) and X is NULL, the initial vector is given internally by the solver (usually randomly generated). |
| n_attempts_max | A value with the maximum number of attempts if the solver does not converge. Each attempt uses a different initial vector. |
| returnqp | Logical. If it is TRUE, it returns the quadratic problems (objective function and constraints). |
| parallel | Logical, if TRUE, the DMUs are computed in parallel (default FALSE). |
| ... | Other parameters to be passed to the solver. |

## Value

A list with the results for the evaluated DMUs and other parameters for reproducibility.

### Note

The model in this function is the "max" version of the model in Cooper et al. (1998), in the sense that maximizes the sum of positive slacks like the conventional additive model in Cooper et al. (1985). Hence, a DMU is $\alpha$-stochastically efficient if and only if the optimal objective value of the problem, (`objval`), is zero (or less than zero).

### Author(s)

**Vicente Bolós** (<vicente.bolos@uv.es>). *Department of Business Mathematics*

**Rafael Benítez** (<rafael.suarez@uv.es>). *Department of Business Mathematics*

**Vicente Coll-Serrano** (<vicente.coll@uv.es>). *Quantitative Methods for Measuring Culture (MC2). Applied Economics.*

University of Valencia (Spain)

### References

Charnes, A.; Cooper, W.W.; Golany, B.; Seiford, L.; Stuz, J. (1985) "Foundations of Data Envelopment Analysis for Pareto-Koopmans Efficient Empirical Production Functions", Journal of Econometrics, 30(1-2), 91-107. doi:10.1016/03044076(85)901332

Cooper, W.W.; Huang, Z.; Lelas, V.; Li, S.X.; Olesen, O.B. (1998) "Chance Constrained Programming Formulations for Stochastic Characterizations of Efficiency and Dominance in DEA", Journal of Productivity Analysis, 9, 53–79. doi:10.1023/A:1018320430249

Cooper, W.W.; Park, K.S.; Pastor, J.T. (1999). "RAM: A Range Adjusted Measure of Inefficiencies for Use with Additive Models, and Relations to Other Models and Measures in DEA". Journal of Productivity Analysis, 11, p. 5-42. doi:10.1023/A:1007701304281

---

## `modelstoch_addsupereff` *Chance Constrained Super-efficiency Additive E-model.*

---

### Description

It solves the chance constrained super-efficiency additive E-model based on the deterministic super-efficiency additive model from Du et al. (2010), under constant and non-constant returns to scale. Besides, the user can set weights for the input and/or output slacks.

We consider $\mathcal{D} = \{\text{DMU}_1, \ldots, \text{DMU}_n\}$ a set of $n$ DMUs with $m$ stochastic inputs and $s$ stochastic outputs. Matrices $\tilde{X} = (\tilde{x}_{ij})$ and $\tilde{Y} = (\tilde{y}_{rj})$ are the input and output data matrices, respectively, where $\tilde{x}_{ij}$ and $\tilde{y}_{rj}$ represent the $i$-th input and $r$-th output of the $j$-th DMU. Moreover, we denote by $X = (x_{ij})$ and $Y = (y_{rj})$ their expected values. In general, we denote vectors by bold-face letters and they are considered as column vectors unless otherwise stated. The 0-vector is denoted by $\mathbf{0}$ and the context determines its dimension.

Given $0 < \alpha < 1$, the program for $\text{DMU}_o$ with constant returns to scale is given by

$$\min_{\boldsymbol{\lambda}, \mathbf{t}^-, \mathbf{t}^+} \quad \mathbf{w}^- \mathbf{t}^- + \mathbf{w}^+ \mathbf{t}^+$$

$$\text{s.t.} \quad P\left\{\left(\tilde{\mathbf{x}}_o - \tilde{X}_{-o}\boldsymbol{\lambda} + \mathbf{t}^-\right)_i \geq 0\right\} \geq 1-\alpha, \qquad i = 1,\ldots,m,$$
$$P\left\{\left(\tilde{Y}_{-o}\boldsymbol{\lambda} - \tilde{\mathbf{y}}_o + \mathbf{t}^+\right)_r \geq 0\right\} \geq 1-\alpha, \qquad r = 1,\ldots,s,$$
$$\boldsymbol{\lambda} \geq \mathbf{0},\ \mathbf{t}^- \geq \mathbf{0},\ \mathbf{t}^+ \geq \mathbf{0},$$

where $\tilde{X}_{-o}, \tilde{Y}_{-o}$ are the input and output data matrices, respectively, defined by $\mathcal{D} - \{\text{DMU}_o\}$, $\boldsymbol{\lambda} = (\lambda_1,\ldots,\lambda_{o-1},\lambda_{o+1},\ldots,\lambda_n)^\top$, $\tilde{\mathbf{x}}_o = (\tilde{x}_{1o},\ldots,\tilde{x}_{mo})^\top$ and $\tilde{\mathbf{y}}_o = (\tilde{y}_{1o},\ldots,\tilde{y}_{so})^\top$ are column vectors. Moreover, $\mathbf{s}^-, \mathbf{s}^+$ are column vectors with the slacks, and $\mathbf{w}^-, \mathbf{w}^+$ are positive row vectors with the weights for the slacks. Different returns to scale can be easily considered by adding the corresponding constraints: $\mathbf{e}\boldsymbol{\lambda} = 1$ (VRS), $0 \leq \mathbf{e}\boldsymbol{\lambda} \leq 1$ (NIRS), $\mathbf{e}\boldsymbol{\lambda} \geq 1$ (NDRS) or $L \leq \mathbf{e}\boldsymbol{\lambda} \leq U$ (GRS), with $0 \leq L \leq 1$ and $U \geq 1$, where $\mathbf{e} = (1,\ldots,1)$ is a row vector.

The deterministic equivalent for a multivariate normal distribution of inputs/outputs is given by

$$\min_{\boldsymbol{\lambda},\mathbf{t}^-,\mathbf{t}^+} \quad \mathbf{w}^-\mathbf{t}^- + \mathbf{w}^+\mathbf{t}^+$$
$$\text{s.t.} \quad \mathbf{x}_o - X_{-o}\boldsymbol{\lambda} + \mathbf{t}^- + \Phi^{-1}(\alpha)\boldsymbol{\sigma}^-(\boldsymbol{\lambda}) \geq \mathbf{0},$$
$$Y_{-o}\boldsymbol{\lambda} - \mathbf{y}_o + \mathbf{t}^+ + \Phi^{-1}(\alpha)\boldsymbol{\sigma}^+(\boldsymbol{\lambda}) \geq \mathbf{0},$$
$$\boldsymbol{\lambda} \geq \mathbf{0},\ \mathbf{t}^- \geq \mathbf{0},\ \mathbf{t}^+ \geq \mathbf{0},$$

where $\Phi$ is the standard normal distribution, and

$$\left(\sigma_i^-(\boldsymbol{\lambda})\right)^2 = \sum_{\substack{j,q=1\\ j,q\neq o}}^{n} \lambda_j\lambda_q \text{Cov}(\tilde{x}_{ij},\tilde{x}_{iq}) - 2\sum_{\substack{j=1\\ j\neq o}}^{n} \lambda_j \text{Cov}(\tilde{x}_{ij},\tilde{x}_{io}) + \text{Var}(\tilde{x}_{io}), \quad i = 1,\ldots,m,$$
$$\left(\sigma_r^+(\boldsymbol{\lambda})\right)^2 = \sum_{\substack{j,q=1\\ j,q\neq o}}^{n} \lambda_j\lambda_q \text{Cov}(\tilde{y}_{rj},\tilde{y}_{rq}) - 2\sum_{\substack{j=1\\ j\neq o}}^{n} \lambda_j \text{Cov}(\tilde{y}_{rj},\tilde{y}_{ro}) + \text{Var}(\tilde{y}_{ro}), \quad r = 1,\ldots,s.$$

## Usage

```
modelstoch_addsupereff(datadea,
                       alpha = 0.05,
                       dmu_eval = NULL,
                       dmu_ref = NULL,
                       orientation = NULL,
                       weight_slack_i = NULL,
                       weight_slack_o = NULL,
                       rts = c("crs", "vrs", "nirs", "ndrs", "grs"),
                       L = 1,
                       U = 1,
                       solver = c("alabama"),
                       X = NULL,
                       n_attempts_max = 5,
                       compute_target = TRUE,
                       returnqp = FALSE,
                       parallel = FALSE,
                       ...)
```

## Arguments

| | |
|---|---|
| datadea | The data of class deadata_stoch, including n DMUs, and the expected values of m inputs and s outputs. |
| alpha | A value for parameter alpha. |
| dmu_eval | A numeric vector containing which DMUs have to be evaluated. If NULL (default), all DMUs are considered. |
| dmu_ref | A numeric vector containing which DMUs are the evaluation reference set. If NULL (default), all DMUs are considered. |
| orientation | This parameter is either NULL (default) or a string, equal to "io" (input-oriented) or "oo" (output-oriented). It is used to modify the weight slacks. If input-oriented, weight_slack_o are taken 0. If output-oriented, weight_slack_i are taken 0. |
| weight_slack_i | A value, vector of length m, or matrix m x ne (where ne is the length of dmu_eval) with the weights of the input super-slacks (t_input). If 0, output-oriented. If weight_slack_i is the matrix of the inverses of inputs of DMUS in dmu_eval (default), the model is unit invariant. |
| weight_slack_o | A value, vector of length s, or matrix s x ne (where ne is the length of dmu_eval) with the weights of the output super-slacks (t_output). If 0, input-oriented. If weight_slack_o is the matrix of the inverses of outputs of DMUS in dmu_eval (default), the model is unit invariant. |
| rts | A string, determining the type of returns to scale, equal to "crs" (constant), "vrs" (variable), "nirs" (non-increasing), "ndrs" (non-decreasing) or "grs" (generalized). |
| L | Lower bound for the generalized returns to scale (grs). |
| U | Upper bound for the generalized returns to scale (grs). |
| solver | Character string with the name of the solver used by function solvecop from package optiSolve. Only alabama solver is currently operational, but more solvers could be implemented in the future. |
| X | Initial vector for the solver in the first attempt. The variables are lambdas of reference DMUs, input t, output t, input sigmas and output sigmas. It can be a matrix of ne rows (where ne is the length of dmu_eval), with different initial vectors for each evaluated DMU. If it is a vector, the same initial vector is applied to all evaluated DMUs. |
| n_attempts_max | A value with the maximum number of attempts if the solver does not converge. Each attempt uses a different initial vector. |
| compute_target | Logical. If it is TRUE, it computes targets, projections and slacks. |
| returnqp | Logical. If it is TRUE, it returns the quadratic problems (objective function and constraints). |
| parallel | Logical, if TRUE, the DMUs are computed in parallel (default FALSE). |
| ... | Other parameters to be passed to the solver. |

## Value

A list with the results for the evaluated DMUs and other parameters for reproducibility.

## Note

A DMU is $\alpha$-stochastically efficient if and only if the optimal objective value of the problem, (`objval`), is zero (or less than zero).

## Author(s)

**Vicente Bolós** (<vicente.bolos@uv.es>). *Department of Business Mathematics*

**Rafael Benítez** (<rafael.suarez@uv.es>). *Department of Business Mathematics*

**Vicente Coll-Serrano** (<vicente.coll@uv.es>). *Quantitative Methods for Measuring Culture (MC2). Applied Economics.*

University of Valencia (Spain)

## References

Du, J.; Liang, L.; Zhu, J. (2010). "A Slacks-based Measure of Super-efficiency in Data Envelopment Analysis. A Comment", European Journal of Operational Research, 204, 694-697. doi:10.1016/j.ejor.2009.12.007

---

`modelstoch_dir` *Chance Constrained Directional Models with Stochastic Directions*

---

## Description

It solves chance constrained directional models with stochastic directions, under constant, variable, non-increasing, non-decreasing or generalized returns to scale. Inputs and outputs must follow a multivariate normal distribution. By default, models are solved in a two-stage process (slacks are maximized).

We consider $\mathcal{D} = \{\text{DMU}_1, \ldots, \text{DMU}_n\}$ a set of $n$ DMUs with $m$ stochastic inputs and $s$ stochastic outputs. Matrices $\tilde{X} = (\tilde{x}_{ij})$ and $\tilde{Y} = (\tilde{y}_{rj})$ are the input and output data matrices, respectively, where $\tilde{x}_{ij}$ and $\tilde{y}_{rj}$ represent the $i$-th input and $r$-th output of the $j$-th DMU. Moreover, we denote by $X = (x_{ij})$ and $Y = (y_{rj})$ their expected values. In general, we denote vectors by bold-face letters and they are considered as column vectors unless otherwise stated. The 0-vector is denoted by $\mathbf{0}$ and the context determines its dimension.

Given $0 < \alpha < 1$, the first stage program for $\text{DMU}_o$ with constant returns to scale is given by

$$\max_{\beta, \boldsymbol{\lambda}} \quad \beta$$

$$\text{s.t.} \quad P\left\{\left(\Theta^-(\beta)\tilde{\mathbf{x}}_o - \tilde{X}\boldsymbol{\lambda}\right)_i \geq 0\right\} \geq 1-\alpha, \quad i = 1, \ldots, m,$$

$$P\left\{\left(\tilde{Y}\boldsymbol{\lambda} - \Theta^+(\beta)\tilde{\mathbf{y}}_o\right)_r \geq 0\right\} \geq 1-\alpha, \quad r = 1, \ldots, s,$$

$$\boldsymbol{\lambda} \geq \mathbf{0},$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)^\top$, $\tilde{\mathbf{x}}_o = (\tilde{x}_{1o}, \ldots, \tilde{x}_{mo})^\top$ and $\tilde{\mathbf{y}}_o = (\tilde{y}_{1o}, \ldots, \tilde{y}_{so})^\top$ are column vectors, $\Theta^-(\beta) = I_m - \beta D^-$, $\Theta^+(\beta) = I_s + \beta D^+$ (with $I_m, I_s$ identity matrices), and $D^- =$

$\text{diag}(d_1^-,\ldots,d_m^-)$, $D^+ = \text{diag}(d_1^+,\ldots,d_s^+)$ are diagonal matrices with orientation parameters $d_1^-,\ldots,d_m^-,d_1^+,\ldots,d_s^+ \geq 0$. Different returns to scale can be easily considered by adding the corresponding constraints: $\mathbf{e}\boldsymbol{\lambda} = 1$ (VRS), $0 \leq \mathbf{e}\boldsymbol{\lambda} \leq 1$ (NIRS), $\mathbf{e}\boldsymbol{\lambda} \geq 1$ (NDRS) or $L \leq \mathbf{e}\boldsymbol{\lambda} \leq U$ (GRS), with $0 \leq L \leq 1$ and $U \geq 1$, where $\mathbf{e} = (1,\ldots,1)$ is a row vector.

The corresponding second stage program is given by

$$\max_{\boldsymbol{\lambda},\mathbf{s}^-,\mathbf{s}^+} \quad \mathbf{w}^-\mathbf{s}^- + \mathbf{w}^+\mathbf{s}^+$$

$$\text{s.t.} \quad P\left\{\left(\Theta^-(\beta^*)\tilde{\mathbf{x}}_o - \tilde{X}\boldsymbol{\lambda} - \mathbf{s}^-\right)_i \geq 0\right\} = 1-\alpha, \quad i = 1,\ldots,m,$$

$$P\left\{\left(\tilde{Y}\boldsymbol{\lambda} - \Theta^+(\beta^*)\tilde{\mathbf{y}}_o - \mathbf{s}^+\right)_r \geq 0\right\} = 1-\alpha, \quad r = 1,\ldots,s,$$

$$\boldsymbol{\lambda} \geq \mathbf{0},\ \mathbf{s}^- \geq \mathbf{0},\ \mathbf{s}^+ \geq \mathbf{0},$$

where $\beta^*$ is the optimal objective function of the first stage program, $\mathbf{s}^-,\mathbf{s}^+$ are column vectors with the slacks, and $\mathbf{w}^-,\mathbf{w}^+$ are positive row vectors with the weights for the slacks.

The deterministic equivalents for a multivariate normal distribution of inputs/outputs are given by

$$\max_{\beta,\boldsymbol{\lambda}} \quad \beta$$

$$\text{s.t.} \quad X\boldsymbol{\lambda} - \Phi^{-1}(\alpha)\boldsymbol{\sigma}^-(\beta,\boldsymbol{\lambda}) \leq \Theta^-(\beta)\mathbf{x}_o,$$

$$Y\boldsymbol{\lambda} + \Phi^{-1}(\alpha)\boldsymbol{\sigma}^+(\beta,\boldsymbol{\lambda}) \geq \Theta^+(\beta)\mathbf{y}_o,$$

$$\boldsymbol{\lambda} \geq \mathbf{0},$$

and for the second stage,

$$\max_{\boldsymbol{\lambda},\mathbf{s}^-,\mathbf{s}^+} \quad \mathbf{w}^-\mathbf{s}^- + \mathbf{w}^+\mathbf{s}^+$$

$$\text{s.t.} \quad X\boldsymbol{\lambda} + \mathbf{s}^- - \Phi^{-1}(\alpha)\boldsymbol{\sigma}^-(\beta^*,\boldsymbol{\lambda}) = \Theta^-(\beta^*)\mathbf{x}_o,$$

$$Y\boldsymbol{\lambda} - \mathbf{s}^+ + \Phi^{-1}(\alpha)\boldsymbol{\sigma}^+(\beta^*,\boldsymbol{\lambda}) = \Theta^+(\beta^*)\mathbf{y}_o,$$

$$\boldsymbol{\lambda} \geq \mathbf{0},\ \mathbf{s}^- \geq \mathbf{0},\ \mathbf{s}^+ \geq \mathbf{0},$$

where $\Phi$ is the standard normal distribution, and

$$\left(\sigma_i^-(\beta,\boldsymbol{\lambda})\right)^2 = \sum_{j,q=1}^{n} \lambda_j\lambda_q \text{Cov}(\tilde{x}_{ij},\tilde{x}_{iq}) - 2(1-\beta d_i^-)\sum_{j=1}^{n}\lambda_j \text{Cov}(\tilde{x}_{ij},\tilde{x}_{io})$$

$$+(1-\beta d_i^-)^2\text{Var}(\tilde{x}_{io}), \quad i = 1,\ldots,m,$$

$$\left(\sigma_r^+(\beta,\boldsymbol{\lambda})\right)^2 = \sum_{j,q=1}^{n} \lambda_j\lambda_q \text{Cov}(\tilde{y}_{rj},\tilde{y}_{rq}) - 2(1+\beta d_r^+)\sum_{j=1}^{n}\lambda_j \text{Cov}(\tilde{y}_{rj},\tilde{y}_{ro})$$

$$+(1+\beta d_r^+)^2\text{Var}(\tilde{y}_{ro}), \quad r = 1,\ldots,s.$$

## Usage

```
modelstoch_dir(datadea,
               alpha = 0.05,
               dmu_eval = NULL,
               dmu_ref = NULL,
               d_input = 1,
               d_output = 1,
               rts = c("crs", "vrs", "nirs", "ndrs", "grs"),
               L = 1,
               U = 1,
               solver = c("alabama"),
               X = NULL,
               give_X = FALSE,
               n_attempts_max = 5,
               maxslack = FALSE,
               weight_slack_i = 1,
               weight_slack_o = 1,
               compute_target = TRUE,
               returnqp = FALSE,
               silent_ud = FALSE,
               parallel = FALSE,
               ...)
```

## Arguments

| | |
|---|---|
| datadea | The data of class deadata_stoch, including n DMUs, and the expected values of m inputs and s outputs. |
| alpha | A value for parameter alpha. |
| dmu_eval | A numeric vector containing which DMUs have to be evaluated. If NULL (default), all DMUs are considered. |
| dmu_ref | A numeric vector containing which DMUs are the evaluation reference set. If NULL (default), all DMUs are considered. |
| d_input | A value, vector of length m, or matrix m x ne (where ne is the length of dmu_eval) with the input orientation parameters. If d_input == 1 (default) and dir_output == 0, it is equivalent to input oriented (beta = 1 - efficiency). |
| d_output | A value, vector of length s, or matrix s x ne (where ne is the length of dmu_eval) with the output orientation parameters. If d_input == 0 and d_output == 1 (default), it is equivalent to output oriented (beta = efficiency - 1). |
| rts | A string, determining the type of returns to scale, equal to "crs" (constant), "vrs" (variable), "nirs" (non-increasing), "ndrs" (non-decreasing) or "grs" (generalized). |
| L | Lower bound for the generalized returns to scale (grs). |
| U | Upper bound for the generalized returns to scale (grs). |
| solver | Character string with the name of the solver used by function solvecop from package optiSolve. Only alabama solver is currently operational, but more solvers could be implemented in the future. |

| | |
|---|---|
| X | Initial vector for the solver in the first attempt. The variables are efficiency, lambdas of reference DMUs, input sigmas and output sigmas. It can be a matrix of ne rows (where ne is the length of dmu_eval), with different initial vectors for each evaluated DMU. If it is a vector, the same initial vector is applied to all evaluated DMUs. |
| give_X | Logical. If it is TRUE and X is NULL, it makes the initial vector to be the variables corresponding to each evaluated DMU. If it is FALSE (default) and X is NULL, the initial vector is given internally by the solver (usually randomly generated). |
| n_attempts_max | A value with the maximum number of attempts if the solver does not converge. Each attempt uses a different initial vector. |
| maxslack | Logical. If it is TRUE, it computes the max slack solution. |
| weight_slack_i | A value, vector of length m, or matrix m x ne (where ne is the length of dmu_eval) with the weights of the input slacks for the max slack solution. |
| weight_slack_o | A value, vector of length s, or matrix s x ne (where ne is the length of dmu_eval) with the weights of the output slacks for the max slack solution. |
| compute_target | Logical. If it is TRUE, it computes targets of the max slack solution. |
| returnqp | Logical. If it is TRUE, it returns the quadratic problems (objective function and constraints) of stage 1. |
| silent_ud | Logical, to avoid warnings related with undesirable variables. |
| parallel | Logical, if TRUE, the DMUs are computed in parallel (default FALSE). |
| ... | Other parameters to be passed to the solver. |

## Value

A list with the results for the evaluated DMUs and other parameters for reproducibility.

## Author(s)

**Vicente Bolós** (<vicente.bolos@uv.es>). *Department of Business Mathematics*

**Rafael Benítez** (<rafael.suarez@uv.es>). *Department of Business Mathematics*

**Vicente Coll-Serrano** (<vicente.coll@uv.es>). *Quantitative Methods for Measuring Culture (MC2). Applied Economics.*

University of Valencia (Spain)

## References

Bolós, V.J.; Benítez, R.; Coll-Serrano, V. (2024). "Chance constrained directional models in stochastic data envelopment analysis", Operations Research Perspectives, 12, 100307.. doi:10.1016/j.orp.2024.100307

## Examples

```
# Example 1.

library(deaR)
data("Coll_Blasco_2006")
```

```
ni <- 2 # number of inputs
no <- 2 # number of outputs
data_example <- make_deadata(datadea = Coll_Blasco_2006,
                             ni = ni,
                             no = no)
nd <- length(data_example$dmunames) # number of DMUs
var_input <- matrix(1, nrow = ni, ncol = nd)
var_output <- matrix(1, nrow = no, ncol = nd)
data_stoch <- make_deadata_stoch(datadea = data_example,
                                 var_input = var_input,
                                 var_output = var_output)
# Evaluate the sixth DMU
Collstochdir <- modelstoch_dir(data_stoch, dmu_eval = 6)
efficiencies(Collstochdir)
```

---

`modelstoch_dir_dd` *Chance Constrained Directional Models with Deterministic Directions*

---

## Description

It solves chance constrained directional models with deterministic directions, under constant, variable, non-increasing, non-decreasing or generalized returns to scale. Inputs and outputs must follow a multivariate normal distribution. By default, models are solved in a two-stage process (slacks are maximized).

We consider $\mathcal{D} = \{\text{DMU}_1, \ldots, \text{DMU}_n\}$ a set of $n$ DMUs with $m$ stochastic inputs and $s$ stochastic outputs. Matrices $\tilde{X} = (\tilde{x}_{ij})$ and $\tilde{Y} = (\tilde{y}_{rj})$ are the input and output data matrices, respectively, where $\tilde{x}_{ij}$ and $\tilde{y}_{rj}$ represent the $i$-th input and $r$-th output of the $j$-th DMU. Moreover, we denote by $X = (x_{ij})$ and $Y = (y_{rj})$ their expected values. In general, we denote vectors by bold-face letters and they are considered as column vectors unless otherwise stated. The 0-vector is denoted by $\mathbf{0}$ and the context determines its dimension.

Given $0 < \alpha < 1$, the first stage program for $\text{DMU}_o$ with constant returns to scale is given by

$$\max_{\beta, \boldsymbol{\lambda}} \quad \beta$$

$$\text{s.t.} \quad P\left\{\left(\tilde{\mathbf{x}}_o - \beta \mathbf{g}^- - \tilde{X}\boldsymbol{\lambda}\right)_i \geq 0\right\} \geq 1 - \alpha, \quad i = 1, \ldots, m,$$

$$P\left\{\left(\tilde{\mathbf{y}}_o + \beta \mathbf{g}^+ - \tilde{Y}\boldsymbol{\lambda}\right)_r \leq 0\right\} \geq 1 - \alpha, \quad r = 1, \ldots, s,$$

$$\boldsymbol{\lambda} \geq \mathbf{0},$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)^\top$, $\tilde{\mathbf{x}}_o = (\tilde{x}_{1o}, \ldots, \tilde{x}_{mo})^\top$ and $\tilde{\mathbf{y}}_o = (\tilde{y}_{1o}, \ldots, \tilde{y}_{so})^\top$ are column vectors, and $\mathbf{g} = (-\mathbf{g}^-, \mathbf{g}^+) \neq \mathbf{0}$ is a preassigned direction (with $\mathbf{g}^- \in \mathbb{R}^m$ and $\mathbf{g}^+ \in \mathbb{R}^s$ non-negative column vectors). Different returns to scale can be easily considered by adding the corresponding

constraints: $\mathbf{e}\boldsymbol{\lambda} = 1$ (VRS), $0 \leq \mathbf{e}\boldsymbol{\lambda} \leq 1$ (NIRS), $\mathbf{e}\boldsymbol{\lambda} \geq 1$ (NDRS) or $L \leq \mathbf{e}\boldsymbol{\lambda} \leq U$ (GRS), with $0 \leq L \leq 1$ and $U \geq 1$, where $\mathbf{e} = (1, \ldots, 1)$ is a row vector.

The corresponding second stage program is given by

$$\max_{\boldsymbol{\lambda}, \mathbf{s}^-, \mathbf{s}^+} \quad \mathbf{w}^- \mathbf{s}^- + \mathbf{w}^+ \mathbf{s}^+$$

$$\text{s.t.} \quad P\left\{\left(\tilde{\mathbf{x}}_o - \beta^* \mathbf{g}^- - \tilde{X}\boldsymbol{\lambda} - \mathbf{s}^-\right)_i \geq 0\right\} \geq 1 - \alpha, \quad i = 1, \ldots, m,$$

$$P\left\{\left(\tilde{\mathbf{y}}_o + \beta^* \mathbf{g}^+ - \tilde{Y}\boldsymbol{\lambda} + \mathbf{s}^+\right)_r \leq 0\right\} \geq 1 - \alpha, \quad r = 1, \ldots, s,$$

$$\boldsymbol{\lambda} \geq \mathbf{0}, \ \mathbf{s}^- \geq \mathbf{0}, \ \mathbf{s}^+ \geq \mathbf{0},$$

where $\beta^*$ is the optimal objective function of the first stage program, $\mathbf{s}^-, \mathbf{s}^+$ are column vectors with the slacks, and $\mathbf{w}^-, \mathbf{w}^+$ are positive row vectors with the weights for the slacks.

The deterministic equivalents for a multivariate normal distribution of inputs/outputs are given by

$$\max_{\beta, \boldsymbol{\lambda}} \quad \beta$$

$$\text{s.t.} \quad \beta \mathbf{g}^- + X\boldsymbol{\lambda} - \Phi^{-1}(\alpha)\boldsymbol{\sigma}^-(\boldsymbol{\lambda}) \leq \mathbf{x}_o,$$

$$-\beta \mathbf{g}^+ + Y\boldsymbol{\lambda} + \Phi^{-1}(\alpha)\boldsymbol{\sigma}^+(\boldsymbol{\lambda}) \geq \mathbf{y}_o,$$

$$\boldsymbol{\lambda} \geq \mathbf{0},$$

and for the second stage,

$$\max_{\boldsymbol{\lambda}, \mathbf{s}^-, \mathbf{s}^+} \quad \mathbf{w}^- \mathbf{s}^- + \mathbf{w}^+ \mathbf{s}^+$$

$$\text{s.t.} \quad \beta^* \mathbf{g}^- + X\boldsymbol{\lambda} + \mathbf{s}^- - \Phi^{-1}(\alpha)\boldsymbol{\sigma}^-(\boldsymbol{\lambda}) = \mathbf{x}_o,$$

$$-\beta^* \mathbf{g}^+ + Y\boldsymbol{\lambda} - \mathbf{s}^+ + \Phi^{-1}(\alpha)\boldsymbol{\sigma}^+(\boldsymbol{\lambda}) = \mathbf{y}_o,$$

$$\boldsymbol{\lambda} \geq \mathbf{0}, \ \mathbf{s}^- \geq \mathbf{0}, \ \mathbf{s}^+ \geq \mathbf{0},$$

where $\Phi$ is the standard normal distribution, and

$$\left(\sigma_i^-(\boldsymbol{\lambda})\right)^2 = \sum_{j,q=1}^{n} \lambda_j \lambda_q \mathrm{Cov}(\tilde{x}_{ij}, \tilde{x}_{iq}) - 2\sum_{j=1}^{n} \lambda_j \mathrm{Cov}(\tilde{x}_{ij}, \tilde{x}_{io})$$

$$+\mathrm{Var}(\tilde{x}_{io}), \quad i = 1, \ldots, m,$$

$$\left(\sigma_r^+(\boldsymbol{\lambda})\right)^2 = \sum_{j,q=1}^{n} \lambda_j \lambda_q \mathrm{Cov}(\tilde{y}_{rj}, \tilde{y}_{rq}) - 2\sum_{j=1}^{n} \lambda_j \mathrm{Cov}(\tilde{y}_{rj}, \tilde{y}_{ro})$$

$$+\mathrm{Var}(\tilde{y}_{ro}), \quad r = 1, \ldots, s.$$

## Usage

```
modelstoch_dir_dd(datadea,
                  alpha = 0.05,
                  dmu_eval = NULL,
                  dmu_ref = NULL,
                  dir_input = NULL,
                  dir_output = NULL,
                  rts = c("crs", "vrs", "nirs", "ndrs", "grs"),
                  L = 1,
                  U = 1,
                  solver = c("alabama"),
                  X = NULL,
                  give_X = FALSE,
                  n_attempts_max = 5,
                  maxslack = FALSE,
                  weight_slack_i = 1,
                  weight_slack_o = 1,
                  compute_target = TRUE,
                  returnqp = FALSE,
                  silent_ud = FALSE,
                  parallel = FALSE,
                  ...)
```

## Arguments

| | |
|---|---|
| datadea | The data of class deadata_stoch, including n DMUs, and the expected values of m inputs and s outputs. |
| alpha | A value for parameter alpha. |
| dmu_eval | A numeric vector containing which DMUs have to be evaluated. If NULL (default), all DMUs are considered. |
| dmu_ref | A numeric vector containing which DMUs are the evaluation reference set. If NULL (default), all DMUs are considered. |
| dir_input | A value, vector of length m, or matrix m x ne (where ne is the length of dmu_eval) with the input directions. If dir_input == input matrix (of DMUS in dmu_eval) and dir_output == 0, it is equivalent to input oriented (beta = 1 - efficiency). If dir_input is omitted, input matrix (of DMUS in dmu_eval) is assigned. |
| dir_output | A value, vector of length s, or matrix s x ne (where ne is the length of dmu_eval) with the output directions. If dir_input == 0 and dir_output == output matrix (of DMUS in dmu_eval), it is equivalent to output oriented (beta = efficiency - 1). If dir_output is omitted, output matrix (of DMUS in dmu_eval) is assigned. |
| rts | A string, determining the type of returns to scale, equal to "crs" (constant), "vrs" (variable), "nirs" (non-increasing), "ndrs" (non-decreasing) or "grs" (generalized). |
| L | Lower bound for the generalized returns to scale (grs). |
| U | Upper bound for the generalized returns to scale (grs). |

| | |
|---|---|
| solver | Character string with the name of the solver used by function solvecop from package optiSolve. Only alabama solver is currently operational, but more solvers could be implemented in the future. |
| X | Initial vector for the solver in the first attempt. The variables are efficiency, lambdas of reference DMUs, input sigmas and output sigmas. It can be a matrix of ne rows (where ne is the length of dmu_eval), with different initial vectors for each evaluated DMU. If it is a vector, the same initial vector is applied to all evaluated DMUs. |
| give_X | Logical. If it is TRUE and X is NULL, it makes the initial vector to be the variables corresponding to each evaluated DMU. If it is FALSE (default) and X is NULL, the initial vector is given internally by the solver (usually randomly generated). |
| n_attempts_max | A value with the maximum number of attempts if the solver does not converge. Each attempt uses a different initial vector. |
| maxslack | Logical. If it is TRUE, it computes the max slack solution. |
| weight_slack_i | A value, vector of length m, or matrix m x ne (where ne is the length of dmu_eval) with the weights of the input slacks for the max slack solution. |
| weight_slack_o | A value, vector of length s, or matrix s x ne (where ne is the length of dmu_eval) with the weights of the output slacks for the max slack solution. |
| compute_target | Logical. If it is TRUE, it computes targets of the max slack solution. |
| returnqp | Logical. If it is TRUE, it returns the quadratic problems (objective function and constraints) of stage 1. |
| silent_ud | Logical, to avoid warnings related with undesirable variables. |
| parallel | Logical, if TRUE, the DMUs are computed in parallel (default FALSE). |
| ... | Other parameters to be passed to the solver. |

## Value

A list with the results for the evaluated DMUs and other parameters for reproducibility.

## Author(s)

**Vicente Bolós** (<vicente.bolos@uv.es>). *Department of Business Mathematics*

**Rafael Benítez** (<rafael.suarez@uv.es>). *Department of Business Mathematics*

**Vicente Coll-Serrano** (<vicente.coll@uv.es>). *Quantitative Methods for Measuring Culture (MC2). Applied Economics.*

University of Valencia (Spain)

## References

Bolós, V.J.; Benítez, R.; Coll-Serrano, V. (2024). "Chance constrained directional models in stochastic data envelopment analysis", Operations Research Perspectives, 12, 100307.. doi:10.1016/j.orp.2024.100307

## Examples

```
# Example 1.

library(deaR)
data("Coll_Blasco_2006")
ni <- 2 # number of inputs
no <- 2 # number of outputs
data_example <- make_deadata(datadea = Coll_Blasco_2006,
                             ni = ni,
                             no = no)
nd <- length(data_example$dmunames) # number of DMUs
var_input <- matrix(1, nrow = ni, ncol = nd)
var_output <- matrix(1, nrow = no, ncol = nd)
data_stoch <- make_deadata_stoch(datadea = data_example,
                                 var_input = var_input,
                                 var_output = var_output)
# Evaluate the sixth DMU
Collstochdirdd <- modelstoch_dir_dd(data_stoch, dmu_eval = 6)
efficiencies(Collstochdirdd)
```

---

`modelstoch_radial` *Chance Constrained Radial Models*

---

## Description

It solves input and output oriented chance constrained radial DEA models under constant (CCR model), variable (BCC model), non-increasing, non-decreasing or generalized returns to scale, based on the model in Cooper et al. (2002). By default, models are solved in a two-stage process (slacks are maximized).

We consider $\mathcal{D} = \{\text{DMU}_1, \ldots, \text{DMU}_n\}$ a set of $n$ DMUs with $m$ stochastic inputs and $s$ stochastic outputs. Matrices $\tilde{X} = (\tilde{x}_{ij})$ and $\tilde{Y} = (\tilde{y}_{rj})$ are the input and output data matrices, respectively, where $\tilde{x}_{ij}$ and $\tilde{y}_{rj}$ represent the $i$-th input and $r$-th output of the $j$-th DMU. Moreover, we denote by $X = (x_{ij})$ and $Y = (y_{rj})$ their expected values. In general, we denote vectors by bold-face letters and they are considered as column vectors unless otherwise stated. The 0-vector is denoted by $\mathbf{0}$ and the context determines its dimension.

Given $0 < \alpha < 1$, the first stage program for $\text{DMU}_o$ with constant returns to scale is given by

$$\min_{\theta, \boldsymbol{\lambda}} \quad \theta$$

$$\text{s.t.} \quad P\left\{\left(\theta \tilde{\mathbf{x}}_o - \tilde{X}\boldsymbol{\lambda}\right)_i \geq 0\right\} \geq 1 - \alpha, \quad i = 1, \ldots, m,$$

$$P\left\{\left(\tilde{Y}\boldsymbol{\lambda} - \tilde{\mathbf{y}}_o\right)_r \geq 0\right\} \geq 1 - \alpha, \quad r = 1, \ldots, s,$$

$$\boldsymbol{\lambda} \geq \mathbf{0},$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)^\top$, $\tilde{\mathbf{x}}_o = (\tilde{x}_{1o}, \ldots, \tilde{x}_{mo})^\top$ and $\tilde{\mathbf{y}}_o = (\tilde{y}_{1o}, \ldots, \tilde{y}_{so})^\top$ are column vectors. Different returns to scale can be easily considered by adding the corresponding constraints: $\mathbf{e}\boldsymbol{\lambda} = 1$ (VRS), $0 \leq \mathbf{e}\boldsymbol{\lambda} \leq 1$ (NIRS), $\mathbf{e}\boldsymbol{\lambda} \geq 1$ (NDRS) or $L \leq \mathbf{e}\boldsymbol{\lambda} \leq U$ (GRS), with $0 \leq L \leq 1$ and $U \geq 1$, where $\mathbf{e} = (1, \ldots, 1)$ is a row vector.

The corresponding second stage program is given by

$$\max_{\boldsymbol{\lambda}, \mathbf{s}^-, \mathbf{s}^+} \quad \mathbf{w}^- \mathbf{s}^- + \mathbf{w}^+ \mathbf{s}^+$$

$$\text{s.t.} \quad P\left\{\left(\theta^* \tilde{\mathbf{x}}_o - \tilde{X}\boldsymbol{\lambda} - \mathbf{s}^-\right)_i \geq 0\right\} = 1 - \alpha, \quad i = 1, \ldots, m,$$

$$P\left\{\left(\tilde{Y}\boldsymbol{\lambda} - \tilde{\mathbf{y}}_o - \mathbf{s}^+\right)_r \geq 0\right\} = 1 - \alpha, \quad r = 1, \ldots, s,$$

$$\boldsymbol{\lambda} \geq \mathbf{0},\ \mathbf{s}^- \geq \mathbf{0},\ \mathbf{s}^+ \geq \mathbf{0},$$

where $\theta^*$ is the optimal objective function of the first stage program, $\mathbf{s}^-, \mathbf{s}^+$ are column vectors with the slacks, and $\mathbf{w}^-, \mathbf{w}^+$ are positive row vectors with the weights for the slacks.

The deterministic equivalents for a multivariate normal distribution of inputs/outputs are given by

$$\min_{\theta, \boldsymbol{\lambda}} \quad \theta$$

$$\text{s.t.} \quad X\boldsymbol{\lambda} - \Phi^{-1}(\alpha)\boldsymbol{\sigma}^-(\theta, \boldsymbol{\lambda}) \leq \theta \mathbf{x}_o,$$

$$Y\boldsymbol{\lambda} + \Phi^{-1}(\alpha)\boldsymbol{\sigma}^+(\boldsymbol{\lambda}) \geq \mathbf{y}_o,$$

$$\boldsymbol{\lambda} \geq \mathbf{0},$$

and for the second stage,

$$\max_{\boldsymbol{\lambda}, \mathbf{s}^-, \mathbf{s}^+} \quad \mathbf{w}^- \mathbf{s}^- + \mathbf{w}^+ \mathbf{s}^+$$

$$\text{s.t.} \quad X\boldsymbol{\lambda} + \mathbf{s}^- - \Phi^{-1}(\alpha)\boldsymbol{\sigma}^-(\theta^*, \boldsymbol{\lambda}) = \theta^* \mathbf{x}_o,$$

$$Y\boldsymbol{\lambda} - \mathbf{s}^+ + \Phi^{-1}(\alpha)\boldsymbol{\sigma}^+(\boldsymbol{\lambda}) = \mathbf{y}_o,$$

$$\boldsymbol{\lambda} \geq \mathbf{0},\ \mathbf{s}^- \geq \mathbf{0},\ \mathbf{s}^+ \geq \mathbf{0},$$

where $\Phi$ is the standard normal distribution, and

$$\left(\sigma_i^-(\theta, \boldsymbol{\lambda})\right)^2 = \sum_{j,q=1}^{n} \lambda_j \lambda_q \mathrm{Cov}(\tilde{x}_{ij}, \tilde{x}_{iq}) - 2\theta \sum_{j=1}^{n} \lambda_j \mathrm{Cov}(\tilde{x}_{ij}, \tilde{x}_{io})$$

$$+\theta^2 \mathrm{Var}(\tilde{x}_{io}), \quad i = 1, \ldots, m,$$

$$\left(\sigma_r^+(\boldsymbol{\lambda})\right)^2 = \sum_{j,q=1}^{n} \lambda_j \lambda_q \mathrm{Cov}(\tilde{y}_{rj}, \tilde{y}_{rq}) - 2 \sum_{j=1}^{n} \lambda_j \mathrm{Cov}(\tilde{y}_{rj}, \tilde{y}_{ro})$$

$$+\mathrm{Var}(\tilde{y}_{ro}), \quad r = 1, \ldots, s.$$

## Usage

```
modelstoch_radial(datadea,
                  alpha = 0.05,
                  dmu_eval = NULL,
                  dmu_ref = NULL,
                  orientation = c("io", "oo"),
                  rts = c("crs", "vrs", "nirs", "ndrs", "grs"),
                  L = 1,
                  U = 1,
                  solver = c("alabama"),
                  X = NULL,
                  give_X = FALSE,
                  n_attempts_max = 5,
                  maxslack = FALSE,
                  weight_slack_i = 1,
                  weight_slack_o = 1,
                  vtrans_i = NULL,
                  vtrans_o = NULL,
                  compute_target = TRUE,
                  returnqp = FALSE,
                  silent_ud = FALSE,
                  parallel = FALSE,
                  ...)
```

## Arguments

| | |
|---|---|
| datadea | The data of class deadata_stoch, including n DMUs, and the expected values of m inputs and s outputs. |
| alpha | A value for parameter alpha. |
| dmu_eval | A numeric vector containing which DMUs have to be evaluated. If NULL (default), all DMUs are considered. |
| dmu_ref | A numeric vector containing which DMUs are the evaluation reference set. If NULL (default), all DMUs are considered. |
| orientation | A string, equal to "io" (input oriented) or "oo" (output oriented). |
| rts | A string, determining the type of returns to scale, equal to "crs" (constant), "vrs" (variable), "nirs" (non-increasing), "ndrs" (non-decreasing) or "grs" (generalized). |
| L | Lower bound for the generalized returns to scale (grs). |
| U | Upper bound for the generalized returns to scale (grs). |
| solver | Character string with the name of the solver used by function solvecop from package optiSolve. Only alabama solver is currently operational, but more solvers could be implemented in the future. |
| X | Initial vector for the solver in the first attempt. The variables are efficiency, lambdas of reference DMUs, input sigmas and output sigmas. It can be a matrix of ne rows (where ne is the length of dmu_eval), with different initial vectors |

| | |
|---|---|
| | for each evaluated DMU. If it is a vector, the same initial vector is applied to all evaluated DMUs. |
| give_X | Logical. If it is TRUE and X is NULL, it makes the initial vector to be the variables corresponding to each evaluated DMU. If it is FALSE (default) and X is NULL, the initial vector is given internally by the solver (usually randomly generated). |
| n_attempts_max | A value with the maximum number of attempts if the solver does not converge. Each attempt uses a different initial vector. |
| maxslack | Logical. If it is TRUE, it computes the max slack solution. |
| weight_slack_i | A value, vector of length m, or matrix m x ne (where ne is the length of dmu_eval) with the weights of the input slacks for the max slack solution. |
| weight_slack_o | A value, vector of length s, or matrix s x ne (where ne is the length of dmu_eval) with the weights of the output slacks for the max slack solution. |
| vtrans_i | Numeric vector of translation for undesirable inputs. If vtrans_i[i] is NA, then it applies the "max + 1" translation to the i-th undesirable input. If vtrans_i is a constant, then it applies the same translation to all undesirable inputs. If vtrans_i is NULL, then it applies the "max + 1" translation to all undesirable inputs. |
| vtrans_o | Numeric vector of translation for undesirable outputs, analogous to vtrans_i, but applied to outputs. |
| compute_target | Logical. If it is TRUE, it computes targets of the max slack solution. |
| returnqp | Logical. If it is TRUE, it returns the quadratic problems (objective function and constraints) of stage 1. |
| silent_ud | Logical, to avoid warnings related with undesirable variables. |
| parallel | Logical, if TRUE, the DMUs are computed in parallel (default FALSE). |
| ... | Other parameters to be passed to the solver. |

## Value

A list with the results for the evaluated DMUs and other parameters for reproducibility.

## Author(s)

**Vicente Bolós** (<vicente.bolos@uv.es>). *Department of Business Mathematics*

**Rafael Benítez** (<rafael.suarez@uv.es>). *Department of Business Mathematics*

**Vicente Coll-Serrano** (<vicente.coll@uv.es>). *Quantitative Methods for Measuring Culture (MC2). Applied Economics.*

University of Valencia (Spain)

## References

Cooper, W.W.; Deng, H.; Huang, Z.; Li, S.X. (2002). "Chance constrained programming approaches to technical efficiencies and inefficiencies in stochastic data envelopment analysis", Journal of the Operational Research Society, 53:12, 1347-1356. doi:10.1057/palgrave.jors.2601433

Land, K.C; Lovell, C.A.K.; Thore, S. (1993). "Chance-constrained data envelopment analysis", Managerial and Decision Economics, Vol. 14, No. 6, 541-554.

## Examples

```
# Example 1.

library(deaR)
data("Coll_Blasco_2006")
ni <- 2 # number of inputs
no <- 2 # number of outputs
data_example <- make_deadata(datadea = Coll_Blasco_2006,
                             ni = ni,
                             no = no)
nd <- length(data_example$dmunames) # number of DMUs
var_input <- matrix(1, nrow = ni, ncol = nd)
var_output <- matrix(1, nrow = no, ncol = nd)
data_stoch <- make_deadata_stoch(datadea = data_example,
                                 var_input = var_input,
                                 var_output = var_output)
# Evaluate the sixth DMU
Collstoch <- modelstoch_radial(data_stoch,
                               dmu_eval = 6)
efficiencies(Collstoch)

# Example 2. Deterministic data with one stochastic input.

library(deaR)
dmunames <- c("A", "B", "C")
nd <- length(dmunames) # Number of DMUs
inputnames <- c("Input_1", "Input_2")
ni <- length(inputnames) # Number of Inputs
outputnames <- c("Output_1", "Output_2", "Output_3")
no <- length(outputnames) # Number of Outputs
X <- matrix(c(5, 14, 8, 15, 7, 12),
            nrow = ni, ncol = nd, dimnames = list(inputnames, dmunames))
Y <- matrix(c(9, 4, 16, 5, 7, 10, 4, 9, 13),
            nrow = no, ncol = nd, dimnames = list(outputnames, dmunames))
datadea <- make_deadata(inputs = X,
                        outputs = Y)
covX <- array(0, dim = c(2, 3, 3))
# The 2nd input is stochastic.
# Since the corresponding 3x3 covariances matrix is symmetric, only values
# above the diagonal are necessary.
covX[2, 1, ] <- c(1.4, 0.9, 0.6)
covX[2, 2, 2:3] <- c(1.5, 0.7)
covX[2, 3, 3] <- 1.2
# Alternatively (note that values below the diagonal are ignored).
covX[2, , ] <- matrix(c(1.4, 0.9, 0.6, 0, 1.5, 0.7, 0, 0, 1.2),
                      nrow = 3,
                      byrow = TRUE)
datadea_stoch <- make_deadata_stoch(datadea,
                                    cov_input = covX)
alpha <- 0.025
res <- modelstoch_radial(datadea_stoch,
                         alpha = alpha,
```

```
                          rts = "vrs")
efficiencies(res)

# Example 3. Replication of results in Land et al. (1993).

library(deaR)
data("PFT1981")
# Selecting DMUs in Program Follow Through (PFT)
PFT <- PFT1981[1:49, ]
PFT <- make_deadata(PFT,
                    inputs = 2:6,
                    outputs = 7:9)
c <- 0.5
var_output <- matrix(c^2, nrow = 3, ncol = 49)
PFT_stoch <- make_deadata_stoch(datadea = PFT,
                                var_output = var_output)
# Evaluate the second DMU
res <- modelstoch_radial(PFT_stoch,
                         dmu_eval = 2)
efficiencies(res)

## Not run:

# Example 3b. Replication of results in Land et al. (1993) parallelized.

library(deaR)
library(doParallel)
data("PFT1981")
# Selecting DMUs in Program Follow Through (PFT)
PFT <- PFT1981[1:49, ]
PFT <- make_deadata(PFT,
                    inputs = 2:6,
                    outputs = 7:9)
c <- 0.5
var_output <- matrix(c^2, nrow = 3, ncol = 49)
PFT_stoch <- make_deadata_stoch(datadea = PFT,
                                var_output = var_output)

# Preparing parallelization
no_cores <- detectCores() - 1
cl <- makeCluster(no_cores)
registerDoParallel(no_cores)
# Evaluate all DMUs
res <- modelstoch_radial(PFT_stoch,
                         parallel = TRUE)
stopImplicitCluster()
efficiencies(res)

## End(Not run)
```

`modelstoch_radial_supereff` *Chance Constrained Radial Super-efficiency Models*

## Description

Solve chance constrained radial super-efficiency DEA models, based on the Cooper et al. (2002) chance constrained radial efficiency models. Analogously to the deterministic case, it removes the evaluated DMU from the set of reference DMUs `dmu_ref` with respect to which it is evaluated.

## Usage

```
modelstoch_radial_supereff(datadea,
                           dmu_eval = NULL,
                           dmu_ref = NULL,
                           parallel = FALSE,
                           ...)
```

## Arguments

| | |
|---|---|
| `datadea` | The data of class `deadata_stoch` with the expected values of inputs and outputs. |
| `dmu_eval` | A numeric vector containing which DMUs have to be evaluated. If `NULL` (default), all DMUs are considered. |
| `dmu_ref` | A numeric vector containing which DMUs are the evaluation reference set. If `NULL` (default), all DMUs are considered. |
| `parallel` | Logical, if `TRUE`, the DMUs are computed in parallel (default `FALSE`). |
| `...` | Model parameters like `orientation` or `rts` to be passed to `modelstoch_radial`, and other parameters to be passed to the solver. |

## Value

A list with the results for the evaluated DMUs and other parameters for reproducibility.

## Note

Radial super-efficiency chance constrained model under non constant (vrs, nirs, ndrs, grs) returns to scale can be unfeasible for certain DMUs.

## Author(s)

**Vicente Bolós** (<vicente.bolos@uv.es>). *Department of Business Mathematics*

**Rafael Benítez** (<rafael.suarez@uv.es>). *Department of Business Mathematics*

**Vicente Coll-Serrano** (<vicente.coll@uv.es>). *Quantitative Methods for Measuring Culture (MC2). Applied Economics.*

University of Valencia (Spain)

## References

Cooper, W.W.; Deng, H.; Huang, Z.; Li, S.X. (2002). “Chance constrained programming approaches to technical efficiencies and inefficiencies in stochastic data envelopment analysis", Journal of the Operational Research Society, 53:12, 1347-1356.

Land, K.C; Lovell, C.A.K.; Thore, S. (1993). "Chance-constrained data envelopment analysis", Managerial and Decision Economics, Vol. 14, No. 6, 541-554.

## See Also

modelstoch_radial

## Examples

```
## Not run:
# Example 1. Replication of results in Land et al. (1993)
# with super-efficiencies and parallelized.

library(deaR)
library(doParallel)
data("PFT1981")
# Selecting DMUs in Program Follow Through (PFT)
PFT <- PFT1981[1:49, ]
PFT <- make_deadata(PFT,
                    inputs = 2:6,
                    outputs = 7:9)
c <- 0.5
var_output <- matrix(c^2, nrow = 3, ncol = 49)
PFT_stoch <- make_deadata_stoch(datadea = PFT,
                                var_output = var_output)
# Preparing parallelization
no_cores <- detectCores() - 1
cl <- makeCluster(no_cores)
registerDoParallel(no_cores)
# Evaluate all DMUs
res <- modelstoch_radial_supereff(PFT_stoch,
                                  parallel = TRUE)
stopImplicitCluster()
efficiencies(res)

## End(Not run)
```

---

# Textile *Data: Cooper et al. (2001).*

---

## Description

Data of textile industries collected from the Statistical Year Book of China of the Chinese Bureau of Statistics. Each year is treated as a DMU.

## Usage

```
data("Textile")
```

## Format

Data frame with 17 rows and 4 columns. Definition of inputs (X) and output (Y):

**X1 = Labor** Expressed in units of 1000 persons.

**X2 = Capital** Stated in units of 1 million Ren Min Bi (Chinese monetary unit) adjusted to 1991 prices.

**Y = Output** Stated in units of 1 million Ren Min Bi (Chinese monetary unit) adjusted to 1991 prices.

## Author(s)

**Vicente Bolos** (<vicente.bolos@uv.es>). *Department of Business Mathematics*

**Rafael Benitez** (<rafael.suarez@uv.es>). *Department of Business Mathematics*

**Vicente Coll-Serrano** (<vicente.coll@uv.es>). *Quantitative Methods for Measuring Culture (MC2). Applied Economics.*

University of Valencia (Spain)

## Source

Cooper, W.W.; Denga, H.; Gua, B.; Lib, S.; Thrall, R.M. (2001). "Using DEA to improve the management of congestion in Chinese industries (1981-1997)", Socio-Economic Planning Sciences, 35(4), 227-242.

## See Also

make_deadata_stoch, modelstoch_radial

# Index

∗ **datasets**
Automobile, 2
Textile, 34

Automobile, 2

efficiencies.dea_stoch, 3

make_deadata_stoch, *3*, 4, *35*
modelstoch_additive, 9
modelstoch_additive_p, 13
modelstoch_addsupereff, 16
modelstoch_dir, 19
modelstoch_dir_dd, 23
modelstoch_radial, *3*, 27, *34, 35*
modelstoch_radial_supereff, 32

Textile, 34