Package 'pioneeR'

October 25, 2024

Title Productivity and Efficiency Analysis using DEA

Version 0.5.0

Description Measure productivity and efficiency using Data Envelopment Analysis (DEA). Available methods include DEA under different technology assumptions, bootstrapping of efficiency scores and calculation of the Malmquist productivity index. Analyses can be performed either in the console or with the provided 'shiny' app. See Banker, R.; Charnes, A.; Cooper, W.W. (1984) <doi:10.1287/mnsc.30.9.1078>, Färe, R.; Grosskopf, S. (1996) <doi:10.1007/978-94-009-1816-0>.

License GPL-3

URL https://riksrevisjonen.github.io/pioneeR/, https://github.com/Riksrevisjonen/pioneeR

BugReports https://github.com/Riksrevisjonen/pioneeR/issues

Depends R (>= 4.1.0), shiny (>= 1.7.0)

Imports bslib (>= 0.6.0), bsicons, cli (>= 3.6.0), ggplot2 (>= 3.3.0), haven (>= 2.5.0), htmltools, lpSolveAPI (>= 5.5.2), markdown, reactable (>= 0.4.0), readxl, rlang (>= 1.1.0), scales, writexl

Suggests deaR, rmarkdown, knitr, testthat (>= 3.0.0), tibble, withr

Language en-GB

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Author Ove Haugland Jakobsen [aut, cre], Aleksander Eilertsen Valberg [aut], Jan Roar Beckstrøm [ctb], Lars Skaage Engebretsen [ctb], Jonas Månsson [ctb], Joachim Sandnes [ctb], National Audit Office of Norway [cph]

Maintainer Ove Haugland Jakobsen <ohj@mac.com>

Repository CRAN

Date/Publication 2024-10-25 07:40:07 UTC

Contents

`bootstrap_dea` *Bootstrap a DEA model*

Description

Run bootstrap on a DEA model to estimate bias corrected efficiency scores and confidence intervals.

Usage

```
bootstrap_dea(dea, alpha = 0.05, bw_rule = "ucv", iterations = 2000)
```

Arguments

`dea`	An object of class 'pioneer_dea' from `compute_dea()`.
`alpha`	One minus the confidence level required. Default is 0.05.
`bw_rule`	A string with the type of bandwidth rule to be used, or a number with the bandwidth parameter. See details.
`iterations`	The number of bootstrap iterations to perform. Default is 2000.

Details

In order to bootstrap a DEA model, you must first create a model object using the `compute_dea()` function. Note that you currently can only bootstrap models using constant or variable returns to scale (RTS). If you try to bootstrap a model using another RTS, the bootstrap will fail with an error message.

The bandwidth argument can be set to either `ucv` for unbiased cross validation, `silverman` for the Silverman rule, or `scott` for the Scott rule. If you provide a number, this will be used directly as the bandwidth parameter `h`. This can be useful to replicate results where `h` is given, such as Simar & Wilson (1998). For most practical applications of the bootstrap, the default value of unbiased cross validation is sensible.

Value

A list of class `pioneer_bootstrap`.

See Also

`compute_dea()`

Examples

```
# Load example data
fare89 <- deaR::Electric_plants
# Estimate efficiency
mod <- compute_dea(
  data = fare89,
  input = c("Labor", "Fuel", "Capital"),
  output = "Output",
  id = "Plant",
)
# Run bootstrap. Reducing the number of iterations to save processing time
boot <- bootstrap_dea(mod, iterations = 100)
# Print results
print(boot)
# Get summary
summary(boot)
```

---

compute_dea *Compute DEA*

Description

Solve an input or output oriented DEA model under constant (`crs`), variable (`vrs`), non-increasing (`drs`), or non-decreasing (`irs`) returns to scale.

Usage

```
compute_dea(
  data,
  input,
  output,
  id = NULL,
  rts = c("crs", "vrs", "drs", "irs"),
  orientation = c("in", "out"),
  super = FALSE,
  slack = FALSE,
  peers = FALSE
)
```

Arguments

`data`	Dataset to analyse.
`input`	A character vector with input variables.
`output`	A character vector with output variables.
`id`	Optional. A string with the DMU id or name variable. Defaults to the rownames of the dataset.
`rts`	Returns to scale.

crs	Constant returns to scale, convexity and free disposability.
vrs	Variable returns to scale, convexity and free disposability.
drs	Decreasing returns to scale, convexity, down-scaling and free disposability.
irs	Increasing returns to scale, (up-scaling, but not down-scaling), convexity and free disposability.

`orientation`	Model orientation.
`super`	If `TRUE` super efficiency scores are calculated.
`slack`	If `TRUE` slack values are calculated.
`peers`	If `TRUE` peers are added to the response.

Value

A list of class `pioneer_dea`.

Examples

```
# Load example data
fare89 <- deaR::Electric_plants
# Estimate efficiency
mod <- compute_dea(
  data = fare89,
  input = c("Labor", "Fuel", "Capital"),
  output = "Output",
  id = "Plant",
  rts = "vrs",
  orientation = "in"
)
# Print results
print(mod)
# Get summary
summary(mod)
# Convert to data frame
df <- as.data.frame(mod)
```

`compute_malmquist` *Compute Malmquist*

Description

Calculate the Malmquist productivity index and its components using Data Envelopment Analysis.

Usage

```
compute_malmquist(data, input, output, id, time, orientation = c("in", "out"))
```

Arguments

`data`	Dataset to analyse.
`input`	A character vector with input variables.
`output`	A character vector with output variables.
`id`	A string with the DMU id or name variable.
`time`	A string with the time period variable.
`orientation`	Model orientation.

Details

Results are returned *a la* Farrell. This implies that for output-oriented models values above one signify improvements in productivity, while values less than one imply deterioration in productivity. For input-oriented models the interpretation is reversed; values less than one denote improvements and values above one denote deterioration.

Note that `compute_malmquist()` only works for balanced panel datasets.

Value

A list of class `pioneer_mlm`

References

Färe, R., Grosskopf, S. (1996). *Intertemporal production frontiers: With dynamic DEA*. Springer.

Examples

```
# Load example data
chnEconomy <- deaR::EconomyLong
# Estimate Malmquist
mod <- compute_malmquist(
  data = chnEconomy,
  id = 'DMUs',
  time = 'Period',
  input = c('Labor', 'Capital'),
  output = 'GIOV',
```

```
  orientation = 'in')
# Print results
print(mod)
# Get summary
summary(mod)
# Convert to data frame
df <- as.data.frame(mod)
```

`compute_scale_efficiency` *Calculate scale efficiency*

Description

Calculate scale efficiency from a set of inputs and outputs and return a data.frame

Usage

```
compute_scale_efficiency(x, y, orientation = c("in", "out"), digits = NULL)
```

Arguments

`x`	A matrix of inputs, created with `create_matrix`
`y`	A matrix of outputs, created with `create_matrix`
`orientation`	`in` for input oriented models or `out` for output oriented
`digits`	An integer with the number of digits to round to. If `NULL` the values are kept unrounded.

Value

A data frame containing the efficiency scores for CRS, VRS, the Scale Efficiency, the VRS to NIRS ratio, and a recommendation on whether to increase or decrease the size of the DMU.

Examples

```
# Create matrices with random values
inputs <- matrix(runif(10, 1, 10), ncol = 2)
outputs <- matrix(runif(10, 1, 10), ncol = 2)
# Compute scale efficiency
res <- compute_scale_efficiency(
  inputs, outputs, orientation = 'out', digits = 2)
```

create_matrix *Create a matrix for input or output variables*

Description

Create a matrix for input or output variables that can be used in DEA models from a supplied data.frame

Usage

```
create_matrix(df, columns, id, normalize = FALSE)
```

Arguments

df	A data.frame containing the data.
columns	A character vector of column names to include in the matrix.
id	A character string specifying the column with DMU IDs.
normalize	A logical indicating whether to normalize the columns by their mean. Defaults to FALSE.

Value

A matrix of inputs or outputs

Examples

```
df <- data.frame(id = 1:3, a = c(10, 20, 30), b = c(5, 15, 25))
create_matrix(df, columns = c("a", "b"), id = "id", normalize = TRUE)
```

run_pioneer *Run pioneeR*

Description

Run the pioneeR app on your local machine.

Usage

```
run_pioneer(x = NULL, port = NULL, ...)
```

Arguments

x	A data frame that should be loaded with the app. See details.
port	Integer. The TCP port that the application should listen on.
...	Other arguments to send to shiny::runApp().

Details

Note that pioneeR must be loaded into the namespace with `library(pioneeR)` before you run the pioneeR app.

You can load a data object in your current environment to the app. You can pass a data frame or a character string with the object name of the data frame you want to be loaded when the app launches. Note that you should only use data frame objects. If you have a tibble (from the tidyverse) or a data table, you can convert to an ordinary data.frame using `as.data.frame()`

Value

None

Examples

```
# Only run this example in interactive R sessions
if (interactive()) {
  df = deaR::Electric_plants
  # Load app with data.frame and set port to 8080
  run_pioneer(x = df, port = 8080)
}
```

`summary_tbl_dea` *Create a summary table for DEA*

Description

Create a binned summary table for efficiency scores from a DEA model.

Usage

```
summary_tbl_dea(x)
```

Arguments

`x`	A numeric vector of efficiency scores or an object of class `pioneer_dea`

Details

The function will return a summary table for efficiency scores from a DEA model. Efficiency scores will be placed in 11 bins, where DMUs with an efficiency score equal to 1 are placed in a separate bin. For output oriented models with range [1, Inf], bins are created with `1/bin`. Bin widths will be equal to models with range [0, 1].

Value

A `data.frame()` with summary statistics

Examples

```
# Load example data
fare89 <- deaR::Electric_plants
# Estimate efficiency
mod <- compute_dea(
  data = fare89,
  input = c("Labor", "Fuel", "Capital"),
  output = "Output",
  rts = "vrs"
)
# Get a summary table of efficiency scores
summary_tbl_dea(mod)
# You can also create the table from a numeric vector of efficiency scores
res <- as.data.frame(mod)
summary_tbl_dea(res$efficiency)
```

`unset_env_vars` *Unset environment variables*

Description

Unsets the environment variables set by pioneeR

Usage

```
unset_env_vars()
```

Value

A logical vector, with elements being `TRUE` if unsetting the variable succeeded

Examples

```
unset_env_vars()
```

Index