Package 'metafrontier'

August 19, 2026

Type Package

Title Analysis of Metafrontier Models for Efficiency and Productivity

Version 0.3.1

Description Implements metafrontier production function models for estimating technical efficiencies and technology gaps for groups of firms that face different restrictions of a common underlying metatechnology (group-specific technologies in the sense of Battese, Rao, and O'Donnell, 2004). Supports both stochastic frontier analysis (SFA) and data envelopment analysis (DEA) based metafrontiers. Includes the deterministic metafrontier of Battese, Rao, and O'Donnell (2004) <doi:10.1023/B:PROD.0000012454.06094.29>, the stochastic metafrontier of Huang, Huang, and Liu (2014) <doi:10.1007/s11123-014-0402-2>, and the metafrontier Malmquist productivity index of O'Donnell, Rao, and Battese (2008) <doi:10.1007/s00181-007-0119-4>. The deterministic metafrontier can be identified by either the minimum sum of absolute deviations (LP) or the minimum sum of squared deviations (QP) criterion. Additional features include panel SFA with time-varying inefficiency, bootstrap confidence intervals for technology gap ratios, a DEA poolability permutation test, latent class metafrontier estimation via the EM algorithm, Murphy-Topel corrected standard errors, convergence diagnostics, import of pre-fitted models from external estimation engines ('sfaR', 'frontier', 'Benchmarking'), and 'ggplot2' visualisation methods.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports stats, graphics, grDevices, Formula, numDeriv, lpSolveAPI, methods

Suggests testthat (>= 3.0.0), knitr, rmarkdown, sfaR, frontier, Benchmarking, quadprog, plm, ggplot2, parallel

VignetteBuilder knitr

URL https://github.com/iik1/metafrontier

BugReports https://github.com/iik1/metafrontier/issues

Config/testthat/edition 3

NeedsCompilation no

Author Erik Enstad [aut, cre] (ORCID: <https://orcid.org/0009-0006-1053-3849>)

Maintainer Erik Enstad <erik.enstad@nhh.no>

Repository CRAN

Date/Publication 2026-08-19 11:40:02 UTC

Contents

`as_metafrontier_model` *Convert a Fitted Frontier Model to Metafrontier Format*

Description

Generic function that extracts the components needed by `metafrontier` from a pre-fitted frontier model. Methods are provided for **sfaR** (`"sfacross"`), **frontier** (`"frontier"`), and **Benchmarking** (`"Farrell"`) objects, as well as plain lists with the required fields.

Usage

```
as_metafrontier_model(x, ...)
```

Arguments

`x`	a fitted frontier model object.
`...`	additional arguments passed to methods.

Details

`metafrontier(models = ...)` calls this function internally on each supplied model, so fitted **sfaR** or **frontier** objects can be passed to `metafrontier()` directly. Manual conversion is only needed for hand-built list models. Converting an object that has already been converted is a no-op, so it is safe to pass converted objects to `metafrontier()` as well.

Note that `Benchmarking::dea()` (`"Farrell"`) objects do not store the inputs, outputs, or frontier coefficients, so the converted model carries only efficiency scores and cannot be used with `metafrontier(models = ...)`; use the formula interface with `method = "dea"` instead. Converting a Farrell object therefore raises a warning.

Value

A list of class `"metafrontier_model"` with components: `beta`, `te`, `X`, `y`, `sigma_v`, `sigma_u`, `logLik`, `hessian`, `n`, `dist`.

Examples

```
# Using a named list:
mod <- as_metafrontier_model(list(
  coefficients = c("(Intercept)" = 2, log_x1 = 0.5, log_x2 = 0.3),
  efficiency = runif(50, 0.7, 1),
  X = matrix(rnorm(150), 50, 3),
  y = rnorm(50, 5)
))
str(mod)
```

autoplot.boot_tgr *Autoplot Method for Bootstrap TGR Objects*

Description

Autoplot Method for Bootstrap TGR Objects

Usage

```
## S3 method for class 'boot_tgr'
autoplot(object, which = c("distribution", "ci"), ...)
```

Arguments

object	a "boot_tgr" object.
which	character. "distribution" (default) or "ci".
...	additional arguments.

Value

A ggplot object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  sim <- simulate_metafrontier(n_groups = 2, n_per_group = 50, seed = 42)
  fit <- metafrontier(log_y ~ log_x1 + log_x2, data = sim$data,
                      group = "group", meta_type = "stochastic")
  boot <- boot_tgr(fit, R = 50, seed = 1, progress = FALSE)
  ggplot2::autoplot(boot)
}
```

autoplot.malmquist_meta *Autoplot Method for Malmquist Meta Objects*

Description

Autoplot Method for Malmquist Meta Objects

Usage

```
## S3 method for class 'malmquist_meta'
autoplot(object, which = c("decomposition", "tgr_evolution", "mpi_trend"), ...)
```

Arguments

object	a "malmquist_meta" object.
which	character. Which plot: "decomposition" (default), "tgr_evolution", or "mpi_trend".
...	additional arguments.

Value

A ggplot object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  panels <- lapply(1:3, function(t) {
    sim <- simulate_metafrontier(n_groups = 2, n_per_group = 30,
                                 seed = 42 + t)
    sim$data$time <- t
    sim$data$id <- seq_len(nrow(sim$data))
    sim$data
  })
  pdata <- do.call(rbind, panels)
  malm <- malmquist_meta(log_y ~ log_x1 + log_x2, data = pdata,
                         group = "group", time = "time")
  ggplot2::autoplot(malm, which = "decomposition")
}
```

---

autoplot.metafrontier *Autoplot Method for Metafrontier Objects*

Description

Creates diagnostic and summary plots for metafrontier objects using **ggplot2**.

Usage

```
## S3 method for class 'metafrontier'
autoplot(
  object,
  which = c("tgr", "efficiency", "decomposition", "frontier"),
  ...
)
```

Arguments

object	a "metafrontier" object.
which	character. Which plot to produce: "tgr" (default) for TGR density distributions by group (degenerate groups with zero-variance TGR are annotated and excluded from the density), "efficiency" for the TE/TGR/TE* decomposition (boxplots), "decomposition" for grouped bars of mean TE, TGR, and TE*, "frontier" for metafrontier vs group frontiers scatter plot.
...	additional arguments (currently unused).

Value

A ggplot object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  sim <- simulate_metafrontier(n_groups = 2, n_per_group = 50, seed = 42)
  fit <- metafrontier(log_y ~ log_x1 + log_x2, data = sim$data,
                      group = "group")
  ggplot2::autoplot(fit, which = "tgr")
  ggplot2::autoplot(fit, which = "decomposition")
}
```

---

boot_tgr *Bootstrap Confidence Intervals for the Technology Gap Ratio*

Description

Computes bootstrap confidence intervals for TGR estimates from a fitted metafrontier model. Supports both parametric (residual resampling) and nonparametric (case resampling) bootstraps.

Usage

```
boot_tgr(
  object,
  R = 999,
  type = c("parametric", "nonparametric"),
  level = 0.95,
  ci_type = c("percentile", "bca"),
  seed = NULL,
  progress = TRUE,
  ncores = 1L,
  ...
)
```

Arguments

object	a "metafrontier" object.
R	integer. Number of bootstrap replications (default 999).
type	character. "parametric" resamples from estimated error distributions; "nonparametric" resamples rows within groups with replacement.
level	numeric. Confidence level (default 0.95).
ci_type	character. "percentile" (default) or "bca" (bias-corrected and accelerated).
seed	optional integer seed for reproducibility.
progress	logical. Show progress bar (default TRUE).
ncores	integer. Number of CPU cores for parallel bootstrap (default 1, sequential). Requires the parallel package.
...	additional arguments passed to metafrontier.

Value

An object of class "boot_tgr" containing:

tgr_boot R x n matrix of bootstrapped TGR values

tgr_original original TGR estimates

ci n x 2 matrix of observation-level confidence intervals

ci_group data frame of group-level mean TGR intervals

R_effective number of successful replications

R requested number of replications

type bootstrap type used

ci_type CI type used

level confidence level

Examples

```
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 100,
                             seed = 42)
fit <- metafrontier(log_y ~ log_x1 + log_x2,
                    data = sim$data, group = "group",
                    meta_type = "stochastic")
boot <- boot_tgr(fit, R = 50, seed = 1)
print(boot)
confint(boot)
```

check_convergence *Check Convergence of All Estimation Stages*

Description

Reports the convergence status of every estimation stage of a fitted metafrontier model: each group-level frontier and the metafrontier itself. This makes it easy to verify that all optimisers (MLE) and mathematical programmes (LP/QP) finished successfully before interpreting technology gap ratios, confidence intervals, or efficiency decompositions.

Usage

```
check_convergence(object, ...)

## S3 method for class 'metafrontier'
check_convergence(object, ...)

## S3 method for class 'lc_metafrontier'
check_convergence(object, ...)

## S3 method for class 'malmquist_meta'
check_convergence(object, ...)
```

Arguments

`object`	a fitted model object.
`...`	additional arguments passed to methods.

Details

For groups supplied via the `models` argument of `metafrontier` the convergence status of the external fitter is not tracked, and the corresponding rows carry `NA` with an explanatory note.

Value

A data frame of class `"metafrontier_convergence"` with one row per estimation stage and columns:

stage stage label, e.g. `"group: G1"` or `"metafrontier"`

method how the stage was estimated: `"MLE"`, `"LP"`, `"QP"`, `"QP (barrier)"`, `"DEA"`, or `"external"`

code the integer convergence code returned by the optimiser (0 indicates success); `NA` for DEA stages and externally fitted groups

converged logical convergence indicator; for DEA stages `TRUE` unless any efficiency score is `NA` (infeasible programme); `NA` for externally fitted groups

note additional detail, e.g. the number of infeasible DEA programmes

Examples

```
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 50, seed = 42)
fit <- metafrontier(log_y ~ log_x1 + log_x2, data = sim$data,
                    group = "group")
check_convergence(fit)
```

`coef.metafrontier` *Extract Coefficients from a Metafrontier Model*

Description

Extract Coefficients from a Metafrontier Model

Usage

```
## S3 method for class 'metafrontier'
coef(object, which = c("meta", "group"), extraPar = FALSE, ...)
```

Arguments

`object`	a `"metafrontier"` object.
`which`	character. `"meta"` (default) returns the metafrontier coefficients; `"group"` returns a named list of group-specific coefficient vectors.
`extraPar`	logical. If `TRUE`, auxiliary parameters are included alongside the frontier coefficients. For `which = "group"` the variance parameters are back-transformed to their natural scale (`sigmaV`, `sigmaU`), `mu` and `eta` are kept as estimated, and heteroscedastic Z coefficients are labelled with their column names. For `which = "meta"` the Stage 2 variance parameters are appended for stochastic metafrontiers.
`...`	additional arguments (currently unused).

Value

A named numeric vector (`which = "meta"`) or a named list of numeric vectors (`which = "group"`).

Examples

```
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 50, seed = 42)
fit <- metafrontier(log_y ~ log_x1 + log_x2, data = sim$data,
                    group = "group", meta_type = "stochastic")
coef(fit)
coef(fit, extraPar = TRUE)
coef(fit, which = "group", extraPar = TRUE)
```

`confint.metafrontier` *Confidence Intervals for Metafrontier Coefficients*

Description

Computes Wald-type confidence intervals for the metafrontier coefficients using the variance-covariance matrix from the stochastic metafrontier.

Usage

```
## S3 method for class 'metafrontier'
confint(
  object,
  parm,
  level = 0.95,
  correction = c("none", "murphy-topel"),
  ...
)
```

Arguments

`object`	a `"metafrontier"` object.
`parm`	a character or integer vector of parameter names or indices. If missing, all frontier coefficients are used.
`level`	the confidence level (default 0.95).
`correction`	character. Passed to `vcov.metafrontier`. `"none"` (default) uses the Stage 2 variance-covariance matrix; `"murphy-topel"` applies the Murphy and Topel (1985) correction for first-stage estimation uncertainty.
`...`	additional arguments (currently unused).

Value

A matrix with columns for the lower and upper bounds.

Examples

```
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 50, seed = 42)
fit <- metafrontier(log_y ~ log_x1 + log_x2, data = sim$data,
                    group = "group", meta_type = "stochastic")
confint(fit)
# With Murphy-Topel correction (requires numDeriv):

confint(fit, correction = "murphy-topel")
```

`efficiencies` *Extract Efficiency Scores from a Metafrontier Model*

Description

Extracts technical efficiency scores from a fitted metafrontier model. Returns either group-specific efficiency (TE), metafrontier efficiency ($TE^* = TE \times TGR$), or the technology gap ratio (TGR).

Usage

```
efficiencies(object, ...)

## S3 method for class 'metafrontier'
efficiencies(object, type = c("meta", "group", "tgr"), estimator = NULL, ...)
```

Arguments

`object`	a fitted `"metafrontier"` object.
`...`	additional arguments (currently unused).
`type`	character. The type of efficiency to return: `"group"` for efficiency relative to the group frontier, `"meta"` (default) for efficiency relative to the metafrontier, or `"tgr"` for the technology gap ratio.
`estimator`	optional character. Override the efficiency estimator used at fit time: `"bc88"` for the Battese-Coelli (1988) conditional expectation $E[\exp(-u)\|\varepsilon]$ or `"jlms"` for $\exp(-E[u\|\varepsilon])$ (Jondrow et al., 1982). Both are stored on SFA fits, so no refitting is needed; `type = "meta"` is recomputed as $TE \times TGR$. The TGR itself does not depend on the estimator. Ignored (with a warning) for DEA fits and externally fitted group models that do not carry both estimators. Default `NULL` returns the scores selected at fit time.

Details

The fundamental metafrontier decomposition is:

$$TE_i^* = TE_i \times TGR_i$$

where TE_i is efficiency relative to the group frontier (returned by `type = "group"`) and TGR_i is the technology gap ratio (returned by `type = "tgr"`).

Value

A numeric vector of efficiency scores of length `nobs(object)`.

See Also

`technology_gap_ratio`, `metafrontier`

Examples

```
set.seed(42)
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 100)
fit <- metafrontier(log_y ~ log_x1 + log_x2,
                    data = sim$data, group = "group")

# Group-level efficiency
te <- efficiencies(fit, type = "group")

# Metafrontier efficiency
te_star <- efficiencies(fit, type = "meta")

# Verify decomposition: TE* = TE x TGR
tgr <- efficiencies(fit, type = "tgr")
all.equal(te_star, te * tgr)
```

---

latent_class_metafrontier

Latent Class Metafrontier

Description

Estimates a metafrontier model where group membership is unobserved, using an EM algorithm to jointly estimate class membership probabilities, class-specific frontier parameters, and the metafrontier.

Usage

```
latent_class_metafrontier(
  formula,
  data,
  n_classes = 2,
  dist = c("hnormal", "tnormal", "exponential"),
  meta_type = c("deterministic", "stochastic"),
  n_starts = 10,
  max_iter = 200,
  tol = 1e-06,
  seed = NULL,
  control = list(),
  ...
)
```

Arguments

formula	a Formula object (y ~ x1 + x2).
data	a data frame.

n_classes	integer. Number of latent classes (default 2).
dist	distribution of the inefficiency term.
meta_type	metafrontier type for Stage 2.
n_starts	integer. Number of random initializations (default 10). The best is selected by log-likelihood.
max_iter	integer. Maximum EM iterations (default 200).
tol	numeric. Convergence tolerance on marginal LL (default 1e-6).
seed	optional integer seed.
control	list of control parameters for the optimiser.
...	additional arguments.

Details

Latent class estimation is available for SFA-based metafrontiers only: the EM posterior class probabilities require a parametric observation-level likelihood, which DEA does not provide. For DEA fits with observed groups, see poolability_test.

The EM algorithm iterates between:

- **E-step**: compute posterior class membership probabilities for each observation using Bayes' rule
- **M-step**: update class-specific frontier parameters via weighted MLE, and update class proportions

Multiple random starts (n_starts) are used to avoid local optima. The run with the highest marginal log-likelihood is selected. After convergence, observations are assigned to classes via MAP (maximum a posteriori), and a standard metafrontier is fitted on the MAP classes.

Value

An object of class "lc_metafrontier" containing:

class_assignment MAP class assignment per observation

posterior n x C matrix of posterior probabilities

pi class mixing proportions

class_params list of class-specific parameter vectors

class_models list of class-specific model summaries

metafrontier the fitted metafrontier object on MAP classes

marginal_ll marginal log-likelihood at convergence

BIC Bayesian Information Criterion

n_classes number of classes

n_iter number of EM iterations used

Examples

```
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 80, seed = 42)
lc <- latent_class_metafrontier(
  log_y ~ log_x1 + log_x2,
  data = sim$data, n_classes = 2, n_starts = 3, seed = 123
)
print(lc)
summary(lc)
coef(lc, which = "meta")
efficiencies(lc, type = "tgr")
```

---

malmquist_meta *Metafrontier Malmquist Productivity Index*

Description

Computes the metafrontier Malmquist total factor productivity (TFP) index and its three-way decomposition into technical efficiency change (TEC), technology gap change (TGC), and metafrontier technical change (TC*) for panel data, following O'Donnell, Rao, and Battese (2008).

Usage

```
malmquist_meta(
  formula = NULL,
  data = NULL,
  group = NULL,
  time = NULL,
  id = NULL,
  method = c("dea", "sfa"),
  dist = c("hnormal", "tnormal", "exponential"),
  estimator = c("bc88", "jlms"),
  orientation = c("output", "input"),
  rts = c("crs", "vrs", "drs", "irs", "fdh"),
  control = list(),
  ...
)
```

Arguments

formula	an object of class Formula. Left-hand side specifies the output(s); right-hand side specifies the inputs. Example: y ~ x1 + x2.
data	a data frame containing all variables, plus the grouping and time variables.
group	a character string naming the column in data that identifies technology groups, or a vector of group indicators.

`time`	a character string naming the column in `data` that identifies time periods, or a vector of time indicators. Periods must be consecutive integers or sortable.
`id`	optional. A character string naming the column in `data` that identifies firms across periods, or a vector of firm identifiers. When supplied, firms are matched across consecutive periods by identifier within each group. When `NULL` (default), firms are matched by row position within each group, which is valid only for balanced panels sorted identically in every period (see Details).
`method`	character. `"dea"` (default) for DEA-based distance functions or `"sfa"` for SFA-based parametric distance functions (an approximation; see Details).
`dist`	character. Distribution of the inefficiency term when `method = "sfa"`: `"hnormal"` (default), `"tnormal"`, or `"exponential"`.
`estimator`	character. Technical efficiency estimator used when `method = "sfa"`: `"bc88"` (default) for the Battese and Coelli (1988) estimator $E[\exp(-u)\|\varepsilon]$, or `"jlms"` for the Jondrow et al. (1982) estimator $\exp(-E[u\|\varepsilon])$. Passed to the group SFA fitter.
`orientation`	character. `"output"` (default) or `"input"`.
`rts`	character. Returns to scale assumption: `"crs"` (default), `"vrs"`, `"drs"`, `"irs"`, or `"fdh"`.
`control`	a list of control parameters for the SFA optimiser.
`...`	additional arguments (currently unused).

Details

The metafrontier Malmquist TFP index decomposes productivity change into three components:

$$M^* = TEC \times TGC \times TC^*$$

where:

- $TEC = TE_{t+1}^{group}/TE_t^{group}$: technical efficiency change relative to the group frontier
- $TGC = TGR_{t+1}/TGR_t$: technology gap change, capturing whether a group's frontier is catching up to or falling behind the metafrontier
- TC^*: metafrontier technical change, measuring the shift of the global production possibility frontier

Firm matching: when `id` is supplied, firms are matched across consecutive periods by identifier within each technology group. Duplicated (id, period) combinations within a group are an error. Observations without a within-group match in the adjacent period, either because the panel is unbalanced or because a firm switches group between periods, are dropped, and a single consolidated warning reports the number dropped per period pair. When `id` is `NULL`, firms are matched by row position within each group; this is valid only for balanced panels sorted identically in every period, so a message is emitted as a reminder, and a warning is issued when group sizes differ across a period pair (the unmatched observations are dropped). Supplying `id` is recommended.

DEA-based computation (`method = "dea"`): for each consecutive pair of periods (s, t), eight sets of LP problems are solved: within-group and pooled efficiencies at each period, plus cross-period

evaluations for the geometric mean formulation of technical change. Distances to the metafrontier are exact distances to the pooled-data frontier, as in O'Donnell, Rao and Battese (2008).

SFA-based computation is an approximation (`method = "sfa"`)**:** period-specific group SFA frontiers are estimated, and each observation's metafrontier distance is approximated by the pointwise maximum of the estimated group frontier functions evaluated at its inputs; no enveloping metafrontier is re-estimated. This coincides with the O'Donnell et al. (2008) metafrontier wherever a single group frontier dominates, but can understate the metafrontier where group frontiers cross, which affects TGC and TC*. Prefer `method = "dea"` when an exact decomposition is required.

Infeasible cross-period programs: under `rts = "vrs"`, `"drs"`, `"irs"`, or `"fdh"`, cross-period LPs can be genuinely infeasible because the reference technology cannot reach the evaluated observation. Such cases yield `NA` (never `Inf`), are excluded from the reported means, and are counted in a single consolidated warning; the counts are stored in the `n_infeasible` and `infeasible_by_period` components. `rts = "crs"` avoids the issue, as does the hyperbolic orientation available in `metafrontier`.

Note that the standard Malmquist index is not a 'proper' (multiplicatively complete and transitive) TFP index in the sense of O'Donnell (2012), so chained comparisons of index levels across more than two periods should be avoided.

Value

An object of class `"malmquist_meta"`, a list with components:

malmquist data frame with columns: `id`, `group`, `period_from`, `period_to`, `MPI` (metafrontier Malmquist TFP index), `TEC` (technical efficiency change), `TGC` (technology gap change), `TC` (metafrontier technical change). The `id` column holds the supplied firm identifiers when `id` is given, and the within-group match position otherwise.

group_malmquist data frame with the within-group Malmquist index decomposition: `MPI_group`, `EC_group`, `TC_group`

meta_malmquist data frame with the metafrontier Malmquist index: `MPI_meta`, `EC_meta`, `TC_meta`

tgr data frame with technology gap ratios at each period endpoint: `id`, `group`, `period_from`, `period_to`, `TGR_from` (TGR at the start period), `TGR_to` (TGR at the end period), and `TGC` (technology gap change, `TGR_to` / `TGR_from`)

call the matched function call

method the estimation method used (`"dea"` or `"sfa"`)

orientation the orientation used

rts the returns to scale assumption

groups group labels

periods time periods

n_infeasible total number of infeasible cross-period DEA programs (always `0` for `method = "sfa"`)

infeasible_by_period data frame with the number of infeasible cross-period DEA programs per period pair (`method = "dea"` only)

References

O'Donnell, C.J., Rao, D.S.P. and Battese, G.E. (2008). Metafrontier frameworks for the study of firm-level efficiencies and technology ratios. *Empirical Economics*, 34(2), 231–255. doi:10.1007/s0018100701194

O'Donnell, C.J. (2012). An aggregate quantity framework for measuring and decomposing productivity change. *Journal of Productivity Analysis*, 38(3), 255–272. doi:10.1007/s1112301202751

Examples

```
# Simulate panel data for 2 groups, 3 time periods
set.seed(42)
panels <- lapply(1:3, function(t) {
  sim <- simulate_metafrontier(
    n_groups = 2, n_per_group = 30,
    tech_gap = c(0, 0.3 + 0.05 * t),
    sigma_u = c(0.2, 0.3),
    seed = 42 + t
  )
  sim$data$time <- t
  sim$data$id <- seq_len(nrow(sim$data))
  sim$data
})
panel_data <- do.call(rbind, panels)

# Compute metafrontier Malmquist index, matching firms by id
malm <- malmquist_meta(
  log_y ~ log_x1 + log_x2,
  data = panel_data,
  group = "group",
  time = "time",
  id = "id"
)
summary(malm)
```

`metafrontier` *Estimate a Metafrontier Production Function*

Description

Estimates group-specific frontiers and a metafrontier that envelops all group technologies. Supports both SFA-based (parametric) and DEA-based (nonparametric) approaches, with deterministic (Battese, Rao, and O'Donnell, 2004) or stochastic (Huang, Huang, and Liu, 2014) metafrontier estimation.

Usage

```
metafrontier(
  formula = NULL,
  data = NULL,
  group = NULL,
  method = c("sfa", "dea"),
  meta_type = c("deterministic", "stochastic"),
  dist = c("hnormal", "tnormal", "exponential"),
  orientation = c("output", "input"),
  rts = c("crs", "vrs", "drs", "irs", "fdh"),
  models = NULL,
  panel = NULL,
  panel_dist = c("bc92", "bc95"),
  type = c("radial", "directional", "hyperbolic"),
  direction = c("proportional", "output", "input"),
  control = list(),
  estimator = c("bc88", "jlms"),
  objective = c("lp", "qp"),
  engine = c("internal", "sfaR", "frontier", "Benchmarking"),
  slack = FALSE,
  ...
)
```

Arguments

formula	an object of class Formula. The left-hand side specifies the (log) output variable. The first right-hand side part specifies inputs for the frontier. An optional second part (separated by |) specifies inefficiency determinants. Example: log_y ~ log_x1 + log_x2 | z1 + z2. Ignored if models is provided.
data	a data frame containing all variables in the formula and the grouping variable. Ignored if models is provided.
group	a character string naming the column in data that identifies technology groups, or a vector of group indicators of length nrow(data). Ignored if models is provided.
method	character. The frontier estimation method for group-specific models: "sfa" (default) for stochastic frontier analysis or "dea" for data envelopment analysis.
meta_type	character. The method for estimating the metafrontier: "deterministic" (default) uses the linear programming approach of Battese, Rao, and O'Donnell (2004); "stochastic" uses the second-stage SFA approach of Huang, Huang, and Liu (2014).
dist	character. Distribution of the one-sided inefficiency term in SFA models. One of "hnormal" (half-normal, default), "tnormal" (truncated normal), or "exponential". Ignored when method = "dea".
orientation	character. For DEA: "output" (default) or "input" orientation. Ignored when method = "sfa".

`rts` character. Returns to scale for DEA: `"crs"` (constant, default), `"vrs"` (variable), `"drs"` (decreasing), `"irs"` (increasing), or `"fdh"` (free disposable hull, i.e. no convexity). Ignored when `method = "sfa"`.

`models` an optional named list of pre-fitted group-specific frontier models (objects from **sfaR** or **frontier**, or hand-built lists). Fitted model objects are converted automatically via `as_metafrontier_model`, so no manual conversion is required (pre-converting is harmless, the conversion is idempotent). Farrell objects from **Benchmarking** store neither coefficients nor data and cannot be used here; use the formula interface with `method = "dea"` instead. If `models` is provided, `formula`, `data`, and `group` are ignored.

`panel` an optional list with components `id` and `time` naming the panel identifier and time columns in `data`. When non-NULL, panel SFA models (BC92/BC95) are used at the group level.

`panel_dist` character. Panel SFA model: `"bc92"` (Battese and Coelli 1992, time-varying inefficiency, default) or `"bc95"` (Battese and Coelli 1995, observation-specific mean). Only used when `panel` is non-NULL.

`type` character. For DEA: `"radial"` (default) for standard radial DEA, `"directional"` for directional distance functions, or `"hyperbolic"` for hyperbolic (graph) efficiency, which contracts inputs and expands outputs simultaneously.

`direction` direction vector for DDF. Either a character preset (`"proportional"` (default), `"output"`, or `"input"`), a numeric vector of length m + s giving a common direction (first m elements for inputs, last s for outputs), or a numeric n x (m + s) matrix of firm-specific directions. With numeric directions the ratio-based TGR is not defined; the additive gap (`ddf_gap`) is reported instead. Only used when `type = "directional"`.

`control` a named list of control parameters passed to `optim`. Common options include `maxit` (maximum iterations, default 5000), `reltol` (relative convergence tolerance, default 1e-10), and `fnscale` (set to -1 internally for maximisation).

`estimator` character. Technical efficiency estimator for SFA models: `"bc88"` (default) for the conditional expectation $E[\exp(-u)|\varepsilon]$ of Battese and Coelli (1988), which is the consistent estimator of technical efficiency, or `"jlms"` for $\exp(-E[u|\varepsilon])$ following Jondrow et al. (1982). Both are stored on the fitted object; see `efficiencies`. Ignored when `method = "dea"`.

`objective` character. Identification criterion for the deterministic metafrontier: `"lp"` (default) minimises the sum of absolute deviations (a linear programme), `"qp"` minimises the sum of squared deviations (a quadratic programme, solved exactly via **quadprog** when available). Both criteria are proposed in Battese, Rao, and O'Donnell (2004). Only used when `method = "sfa"` and `meta_type = "deterministic"`.

`engine` character. Estimation backend for the group frontiers: `"internal"` (default) uses the package's own estimators; `"sfaR"` or `"frontier"` delegate the SFA group frontiers to `sfacross` or `sfa` (cross-sectional, single-part formulas only); `"Benchmarking"` delegates the DEA group frontiers and the pooled metafrontier to `dea` (radial only), using its `XREF`/`YREF` external-reference facility for the metafrontier stage. The metafrontier stage for SFA methods is always estimated internally (the Murphy-Topel correction requires the internal likelihood).

slack	logical. For radial DEA, compute second-stage input and output slacks (with the radial score held fixed) against both the group and the pooled reference sets. Default FALSE.
...	additional arguments passed to the group-level estimation functions.

Details

The metafrontier framework decomposes efficiency relative to a global technology into two components:

$$TE_i^* = TE_i \times TGR_i$$

where TE_i is efficiency relative to the group frontier and TGR_i is the technology gap ratio measuring how close the group frontier is to the metafrontier.

The deterministic metafrontier is identified by one of the two criteria proposed by Battese, Rao, and O'Donnell (2004), subject in both cases to the constraint that the metafrontier envelops all group frontiers: minimising the sum of absolute deviations, which reduces to a linear programme because the envelope constraints force every deviation to be non-negative (O'Donnell, Rao, and Battese, 2008, Eqs. 23-25), or minimising the sum of squared deviations, a convex quadratic programme. The LP (`objective = "lp"`, the default) is solved via **lpSolveAPI**; the QP (`objective = "qp"`) is solved exactly via **quadprog** when available, with an adaptive-barrier fallback via `constrOptim()`. The stochastic metafrontier (Huang, Huang, and Liu, 2014) replaces this with a second-stage SFA, providing a distributional framework for inference on the TGR.

Convergence and failure handling: estimation stops with an error only when no usable estimate exists (for example, when both the BFGS and Nelder-Mead optimisers fail for a group frontier). When an optimiser stops at a non-zero convergence code, the fitted object is returned with a warning and the code is recorded; use `check_convergence` or `summary()` to verify all estimation stages before interpreting technology gap ratios, confidence intervals, or productivity decompositions. Infeasible DEA programmes yield `NA` efficiency scores, accompanied by a warning and counted by `check_convergence`.

Note on standard errors (stochastic metafrontier): The stochastic metafrontier is a two-stage estimator. Stage 2 treats the fitted group frontier values as data, so the reported standard errors, confidence intervals, and variance-covariance matrix do not account for estimation uncertainty from Stage 1 (the generated-regressor problem; see Murphy and Topel, 1985). Use `vcov(fit, correction = "murphy-topel")` or bootstrap-based confidence intervals via `boot_tgr` for corrected inference.

Note on frontier orientation (SFA path): The SFA estimation path assumes a production frontier ($\varepsilon = v - u$). Cost frontiers ($\varepsilon = v + u$) are not currently supported via the SFA path. The DEA path supports both `orientation = "output"` and `orientation = "input"`.

Value

An object of class `"metafrontier"` (with subclass `"metafrontier_sfa"` or `"metafrontier_dea"`), containing:

call the matched function call

group_models list of fitted group-specific models

meta_coef estimated metafrontier parameters

group_coef list of group-specific coefficient vectors

tgr technology gap ratios for each observation

te_group group-specific technical efficiency

te_meta metafrontier technical efficiency (TE* = TE x TGR)

logLik_groups log-likelihoods of group models

nobs number of observations per group and total

groups group labels

method estimation method used

meta_type metafrontier type used

meta_convergence integer convergence code for the metafrontier stage (0 = success; `optim` codes for the stochastic metafrontier and the QP barrier fallback; 0 for a successful LP or DEA solution). Each SFA group model in `group_models` additionally carries its own `convergence` code. Use `check_convergence` to inspect all stages.

estimator, objective, engine, meta_solver the estimation choices used for the fit

References

Battese, G.E., Rao, D.S.P. and O'Donnell, C.J. (2004). A metafrontier production function for estimation of technical efficiencies and technology gaps for firms operating under different technologies. *Journal of Productivity Analysis*, 21(1), 91–103. doi:10.1023/B:PROD.0000012454.06094.29

Huang, C.J., Huang, T.-H. and Liu, N.-H. (2014). A new approach to estimating the metafrontier production function based on a stochastic frontier framework. *Journal of Productivity Analysis*, 42(3), 241–254. doi:10.1007/s1112301404022

O'Donnell, C.J., Rao, D.S.P. and Battese, G.E. (2008). Metafrontier frameworks for the study of firm-level efficiencies and technology ratios. *Empirical Economics*, 34(2), 231–255. doi:10.1007/s0018100701194

Examples

```
# Simulate metafrontier data
set.seed(42)
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 100)

# Estimate deterministic SFA metafrontier (BRO 2004)
fit <- metafrontier(log_y ~ log_x1 + log_x2,
                    data = sim$data,
                    group = "group",
                    method = "sfa",
                    meta_type = "deterministic")
summary(fit)

# Technology gap ratios
tgr <- technology_gap_ratio(fit)
summary(tgr)
```

plot.metafrontier *Plot a Metafrontier Object*

Description

Produces diagnostic and summary plots for a fitted metafrontier model.

Usage

```
## S3 method for class 'metafrontier'
plot(x, which = c("tgr", "efficiency", "decomposition", "frontier"), ...)
```

Arguments

x	a "metafrontier" object.
which	character. Type of plot to produce: "tgr" (default) for TGR distributions by group, "efficiency" for TE* vs TE scatter coloured by group, "decomposition" for side-by-side boxplots of TE, TGR, and TE* by group, or "frontier" for group frontier vs metafrontier (only for single-input SFA models).
...	additional graphical parameters passed to base plotting functions.

Value

Invisibly returns x.

Examples

```
set.seed(42)
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 100)
fit <- metafrontier(log_y ~ log_x1 + log_x2,
                    data = sim$data, group = "group")
plot(fit, which = "tgr")
plot(fit, which = "decomposition")
```

poolability_test *Test Poolability of Group Frontiers*

Description

Tests the null hypothesis that all groups share a common frontier (i.e., the metafrontier coincides with all group frontiers) against the alternative that group-specific frontiers differ. For SFA-based metafrontiers a likelihood ratio test is used; for DEA-based metafrontiers a permutation test is used.

Usage

```
poolability_test(object, B = 199, seed = NULL, ...)
```

Arguments

`object`	a fitted `"metafrontier"` object with `method = "sfa"` or `method = "dea"`.
`B`	integer. Number of permutation replicates for the DEA permutation test (default 199). Ignored for SFA objects.
`seed`	integer or `NULL`. Random seed for the DEA permutation test, for reproducibility. Ignored for SFA objects.
`...`	additional arguments (currently unused).

Details

Likelihood ratio test (SFA). The LR statistic is:

$$LR = -2[LL_{pooled} - \sum_j LL_j]$$

where LL_{pooled} is the log-likelihood of the pooled (single frontier) model and LL_j are the group-specific log-likelihoods. Under H0, the statistic follows a chi-squared distribution with degrees of freedom equal to $df = k_{groups} - k_{pooled}$, where k_{groups} is the total number of parameters across all group-specific models and k_{pooled} is the number of parameters in the pooled model. For J groups each with p frontier parameters plus distributional parameters, this equals $(J-1) \times p_{total}$ where p_{total} includes frontier coefficients, σ_v, and σ_u (and μ for truncated-normal). This test requires a likelihood and is therefore only available for SFA-based metafrontiers.

Permutation test (DEA). DEA has no likelihood, so the poolability hypothesis is assessed by a permutation test. Under the null of a single pooled technology, group labels are exchangeable: reassigning observations to groups at random should not systematically change the distance between the group frontiers and the metafrontier. The observed statistic is the mean technology gap, $S_{obs} = \mathrm{mean}(1 - TGR_i)$, and its null distribution is approximated by refitting the metafrontier on `B` random permutations of the group labels. The p-value is $(1 + \#\{S_b \geq S_{obs}\})/(B+1)$, following the aggregate-efficiency inference logic of Simar and Zelenyuk (2007). The smoothed subsampling approach of Kneip, Simar, and Wilson (2016) is the asymptotically rigorous alternative for testing hypotheses in nonparametric production models; the permutation test offered here is a computationally simple approximation. The default `B = 199` is a pragmatic choice; p-values have resolution $1/(B+1)$, so increase `B` for finer resolution.

Value

A list of class `"htest"` with components:

statistic the test statistic (LR statistic for SFA; the mean technology gap, $\bar{S} = \mathrm{mean}(1 - TGR)$, for DEA)

parameter degrees of freedom (SFA) or the effective number of permutation replicates (DEA)

p.value p-value of the test

method description of the test

References

Simar, L. and Zelenyuk, V. (2007). Statistical inference for aggregates of Farrell-type efficiencies. *Journal of Applied Econometrics*, 22(7), 1367–1394. doi:10.1002/jae.991

Kneip, A., Simar, L. and Wilson, P.W. (2016). Testing hypotheses in nonparametric models of production. *Journal of Business & Economic Statistics*, 34(3), 435–456. doi:10.1080/07350015.2015.1049747

Examples

```
set.seed(42)
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 200,
                             tech_gap = c(0, 0.5))
fit <- metafrontier(log_y ~ log_x1 + log_x2,
                    data = sim$data, group = "group")
poolability_test(fit)


# DEA permutation test
fit_dea <- metafrontier(log_y ~ log_x1 + log_x2,
                        data = sim$data, group = "group",
                        method = "dea")
poolability_test(fit_dea, B = 99, seed = 1)
```

`predict.metafrontier` *Predict Frontier Values from a Metafrontier Model*

Description

Computes predicted frontier values at given input levels using either the metafrontier or a group-specific frontier.

Usage

```
## S3 method for class 'metafrontier'
predict(object, newdata = NULL, type = c("meta", "group"), ...)
```

Arguments

`object`	a `"metafrontier"` object.
`newdata`	an optional data frame of new inputs. If omitted, the training data predictions are returned.
`type`	character. `"meta"` (default) for metafrontier predictions, or `"group"` for group-specific frontier predictions (requires a `group` column in `newdata`).
`...`	additional arguments (currently unused).

Value

A numeric vector of predicted frontier values.

Examples

```
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 50, seed = 42)
fit <- metafrontier(log_y ~ log_x1 + log_x2, data = sim$data,
                    group = "group", meta_type = "stochastic")
pred <- predict(fit)
# Out-of-sample prediction:
newdata <- data.frame(log_x1 = c(1, 2), log_x2 = c(1.5, 2.5))
predict(fit, newdata = newdata)
```

---

print.metafrontier *Print a Metafrontier Object*

Description

Prints a compact overview of a fitted metafrontier model: the estimation method and metafrontier type, the efficiency estimator and identification objective (where applicable), the groups and their sample sizes, group log-likelihoods, mean technology gap ratio by group, and a one-line convergence status.

Usage

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

Arguments

x	a "metafrontier" object.
...	additional arguments (currently unused).

Value

Invisibly returns x.

Examples

```
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 50, seed = 42)
fit <- metafrontier(log_y ~ log_x1 + log_x2, data = sim$data, group = "group")
print(fit)
```

`select_n_classes` *Select Number of Latent Classes via BIC*

Description

Fits a latent class metafrontier model for each value in `n_classes_range` and tabulates the Bayesian information criterion (BIC) and marginal log-likelihood of each fit. The optimal number of classes minimises BIC. Fits that fail are silently dropped from the table. Because the EM algorithm can converge to local optima, the ranking is sensitive to the number of random starts: pass `n_starts` (forwarded to `latent_class_metafrontier`) and increase it for a more reliable comparison across class counts.

Usage

```
select_n_classes(formula, data, n_classes_range = 2:5, ...)
```

Arguments

`formula`	formula.
`data`	data frame.
`n_classes_range`	integer vector of class counts to try.
`...`	additional arguments passed to `latent_class_metafrontier`, notably `n_starts`.

Value

A data frame with columns `n_classes`, `BIC`, and `marginal_ll`.

Examples

```
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 80, seed = 42)
bic_table <- select_n_classes(
  log_y ~ log_x1 + log_x2,
  data = sim$data, n_classes_range = 2:3,
  n_starts = 3, seed = 42
)
print(bic_table)
```

simulate_metafrontier *Simulate Metafrontier Data*

Description

Generates synthetic data from a known metafrontier data-generating process. Each group frontier lies weakly below the metafrontier, consistent with groups facing different restrictions of a common metatechnology (Battese, Rao and O'Donnell, 2004). Useful for Monte Carlo simulations, package testing, and teaching.

Usage

```
simulate_metafrontier(
  n_groups = 2L,
  n_per_group = 100L,
  n_inputs = 2L,
  beta_meta = NULL,
  tech_gap = NULL,
  sigma_u = NULL,
  sigma_v = 0.2,
  seed = NULL,
  beta_groups = NULL,
  input_means = NULL,
  input_corr = NULL
)
```

Arguments

n_groups	integer. Number of technology groups (default 2).
n_per_group	integer or integer vector. Number of observations per group. If a single value, the same number is used for all groups. If a vector, must be of length n_groups.
n_inputs	integer. Number of input variables (default 2).
beta_meta	numeric vector. Metafrontier coefficients (including intercept). Length must be n_inputs + 1. Default: c(1.0, seq(0.5, 0.2, length.out = n_inputs)), i.e. c(1.0, 0.5, 0.2) for the default two inputs. Ignored when beta_groups is supplied.
tech_gap	numeric vector of length n_groups. The technology gap for each group, defined as the reduction in the intercept relative to the metafrontier. Default: evenly spaced from 0 to 0.5. Ignored (with a warning) when beta_groups is supplied.
sigma_u	numeric vector of length n_groups. Standard deviation of the half-normal inefficiency term for each group. Default: rep(0.3, n_groups).
sigma_v	numeric. Standard deviation of the symmetric noise term. Default: 0.2.
seed	integer or NULL. Random seed for reproducibility.

`beta_groups` optional group-specific frontier coefficients, including slopes: either an `n_groups` x (`n_inputs` + 1) numeric matrix (one row per group) or a list of `n_groups` numeric vectors of length `n_inputs` + 1. When supplied, it replaces the intercept-shift construction based on `tech_gap`; see Details. Default `NULL` (intercept-shift design).

`input_means` optional `n_groups` x `n_inputs` numeric matrix of per-group mean log-input levels. When supplied, the log-inputs for group `g` are drawn from a normal distribution centred at `input_means[g, ]`; see Details. Default `NULL` (identical uniform inputs across groups).

`input_corr` optional `n_inputs` x `n_inputs` correlation matrix for the log-inputs. When supplied, the log-inputs are drawn from a multivariate normal distribution with this correlation structure; see Details. Default `NULL` (independent inputs).

Details

By default the group frontiers share the metafrontier slopes and differ only in their intercepts, so the true technology gap ratio (TGR) is constant within each group and equals `exp(-tech_gap[g])`. When `beta_groups` is supplied the group frontiers may differ in their slopes, in which case no single log-linear metafrontier envelops all groups: the tightest log-linear envelope is then a pseudo-true quantity. The returned `true_tgr` is instead computed observation by observation against the pointwise maximum over the group frontiers, $TGR_i = \exp(x_i^\top \beta_g - \max_j x_i^\top \beta_j)$, which is guaranteed to lie in (0, 1]. The true group frontier for each firm is $x_i^\top \beta_g$, `true_te` is generated exactly as in the default design, and `true_te_star` = `true_te` * `true_tgr`. In this case `params$beta_meta` is `NULL` and `params$beta_groups` holds the supplied coefficients.

By default the log-inputs are drawn i.i.d. from a uniform distribution on `[0, 5]`, identically across groups. Supplying `input_means` and/or `input_corr` switches to normal log-inputs with standard deviation `5 / sqrt(12)` (matching the spread of the uniform draws), centred at `input_means[g, ]` (2.5 for every group and input when `input_means` is `NULL`). When `input_corr` is supplied the draws are multivariate normal with that correlation matrix; when it is `NULL` but `input_means` is given, the inputs are drawn independently.

Value

A list with components:

data a data frame with columns `log_y`, `log_x1`, `log_x2`, ..., `group`, and the true underlying values

params a list of the true parameters used for generation

Examples

```
sim <- simulate_metafrontier(n_groups = 3, n_per_group = 200,
                             sigma_u = c(0.2, 0.4, 0.3))
str(sim$data)
table(sim$data$group)

# The true metafrontier coefficients
sim$params$beta_meta

# Group-specific slopes: per-observation true TGR
```

```
sim2 <- simulate_metafrontier(
  beta_groups = rbind(c(1.0, 0.5, 0.2), c(0.9, 0.6, 0.1))
)
range(sim2$data$true_tgr)
```

`simulate_panel_metafrontier` *Simulate Panel Metafrontier Data*

Description

Generates a simulated panel dataset with known metafrontier parameters for Monte Carlo studies and testing. Implements the Battese-Coelli 1992 DGP with time-varying inefficiency.

Usage

```
simulate_panel_metafrontier(
  n_groups = 2,
  n_firms_per_group = 30,
  n_periods = 5,
  beta_meta = c(1, 0.5, 0.3),
  tech_gap = NULL,
  sigma_u = 0.3,
  sigma_v = 0.2,
  eta = 0.05,
  seed = NULL,
  attrition = 0
)
```

Arguments

`n_groups`	integer. Number of technology groups.
`n_firms_per_group`	integer. Number of firms per group.
`n_periods`	integer. Number of time periods.
`beta_meta`	numeric vector. Metafrontier coefficients.
`tech_gap`	numeric vector of group technology gaps.
`sigma_u`	numeric. Standard deviation of the firm effect.
`sigma_v`	numeric. Standard deviation of noise.
`eta`	numeric. Time-decay parameter for BC92.
`seed`	integer or NULL. Random seed.

`attrition` numeric in [0, 0.5]. Probability that each firm-period observation after a firm's first period is dropped independently, producing an unbalanced panel. Every firm's first period is always kept, so all firms remain in the data. The attrition draws are made after all other random numbers, so `attrition = 0` (the default) reproduces legacy balanced datasets exactly for the same seed. The realised share of at-risk observations dropped is stored in `params$attrition_share`.

Value

A list with components:

data data frame with columns: `firm`, `year`, `group`, `log_y`, `log_x1`, `log_x2`, `true_te`, `true_u`, `true_v`

params list of true parameter values used in generation, including `attrition` and the realised `attrition_share`

Examples

```
sim <- simulate_panel_metafrontier(
  n_groups = 2, n_firms_per_group = 20,
  n_periods = 5, seed = 42
)
head(sim$data)
str(sim$params)

# An unbalanced panel with roughly 20% attrition
sim_unbal <- simulate_panel_metafrontier(seed = 42, attrition = 0.2)
table(table(sim_unbal$data$firm))
```

---

summary.metafrontier *Summary of a Metafrontier Model*

Description

Computes group-level summaries of technical efficiency (TE), technology gap ratio (TGR), and metafrontier efficiency (TE*), full coefficient tables for each group frontier (including variance parameters and, for BC92 panels, `eta`, all with standard errors where a Hessian is available), the metafrontier coefficient table (with Murphy-Topel corrected standard errors where applicable), and a per-stage convergence table.

Usage

```
## S3 method for class 'metafrontier'
summary(object, ...)
```

Arguments

`object` a `"metafrontier"` object.

`...` additional arguments (currently unused).

Value

An object of class "summary.metafrontier": a list with components

call the matched call of the original fit

method estimation method ("sfa" or "dea")

meta_type metafrontier type ("deterministic" or "stochastic")

groups character vector of group labels

nobs named vector of observation counts (total and per group)

group_tables named list of coefficient matrices, one per group, with columns Estimate, Std. Error, z value, and Pr(>|z|) where standard errors are available (empty list for DEA fits)

meta_table metafrontier coefficient matrix in the same format, or NULL for DEA fits

tgr_summary data frame of TGR statistics by group, as returned by tgr_summary

efficiency_summary data frame with mean TE, mean TGR, and mean TE* by group

logLik_groups named vector of group log-likelihoods, or NULL

meta_logLik Stage 2 log-likelihood of the stochastic metafrontier, or NULL

convergence data frame with columns stage, code, and converged recording the optimiser status of each estimation stage, or NULL if unavailable; see check_convergence

Examples

```
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 50, seed = 42)
fit <- metafrontier(log_y ~ log_x1 + log_x2, data = sim$data,
                    group = "group", meta_type = "stochastic")
s <- summary(fit)
print(s)
```

---

technology_gap_ratio *Extract Technology Gap Ratios*

Description

Extracts the technology gap ratios (TGR) from a fitted metafrontier model. The TGR measures how close a group's production frontier is to the global metafrontier at each observation's input mix.

Usage

```
technology_gap_ratio(object, by_group = TRUE, ...)
```

Arguments

object	a fitted "metafrontier" object.
by_group	logical. If TRUE (default), returns a named list of TGR vectors, one per group. If FALSE, returns a single numeric vector.
...	additional arguments (currently unused).

Details

The technology gap ratio is defined as:

$$TGR_i = \frac{f(x_i; \hat{\beta}_j)}{f(x_i; \hat{\beta}^*)}$$

for SFA-based metafrontiers, and

$$TGR_i = \frac{TE_i^*}{TE_i^{group}}$$

for DEA-based metafrontiers.

Under the deterministic metafrontier and DEA, TGR lies in (0, 1] by construction. Under the stochastic metafrontier of Huang, Huang, and Liu (2014), TGR can exceed 1 for some observations because the metafrontier need not envelop the group frontiers at every point.

A TGR of 1 means the group frontier coincides with the metafrontier at that input mix. Values less than 1 indicate a technology gap. Since each group technology is a restricted subset of the common metatechnology (Battese, Rao and O'Donnell, 2004), the gap reflects the restrictions a group faces (regulation, environment, endowments) rather than a fundamentally different technology.

Value

If `by_group = TRUE`, a named list of numeric vectors. If `by_group = FALSE`, a numeric vector of length `nobs(object)`.

See Also

`metafrontier`, `efficiencies.metafrontier`

Examples

```
set.seed(42)
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 100)
fit <- metafrontier(log_y ~ log_x1 + log_x2,
                    data = sim$data, group = "group")
tgr <- technology_gap_ratio(fit)
lapply(tgr, summary)
```

---

tgr_summary *Summary of Technology Gap Ratios*

Description

Prints a summary table of TGR statistics by group. The underlying observation-level TGR values are the same as those returned by `efficiencies(object, type = "tgr")`.

Usage

```
tgr_summary(object, ...)
```

Arguments

object	a fitted "metafrontier" object.
...	additional arguments (currently unused).

Value

A data frame with columns: Group, N, Mean, SD, Min, Q1, Median, Q3, Max.

See Also

efficiencies.metafrontier, technology_gap_ratio, boot_tgr

Examples

```
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 50,
                             seed = 42)
fit <- metafrontier(log_y ~ log_x1 + log_x2, data = sim$data,
                    group = "group", method = "sfa",
                    meta_type = "deterministic")
tgr_summary(fit)
```

---

vcov.metafrontier *Variance-Covariance Matrix for Metafrontier Coefficients*

Description

Returns the variance-covariance matrix of the Stage 2 (metafrontier) coefficients, or, with which = "group", the per-group matrices from the Stage 1 maximum likelihood fits. NULL is returned when no Stage 2 Hessian exists: the deterministic metafrontier is fitted by LP/QP optimisation and has no sampling variance in this framework, so which = "meta" returns NULL with a warning; with which = "group", list entries are NULL for groups without a stored Hessian (e.g. externally fitted models). DEA-based metafrontiers are nonparametric and vcov() signals an error; use boot_tgr for inference instead.

Usage

```
## S3 method for class 'metafrontier'
vcov(
  object,
  correction = c("none", "murphy-topel"),
  which = c("meta", "group"),
  extraPar = FALSE,
  ...
)
```

Arguments

object	a "metafrontier" object.
correction	character. "none" (default) returns the Stage 2 variance-covariance matrix. "murphy-topel" applies the Murphy and Topel (1985) correction for first-stage estimation uncertainty (the generated-regressor problem). Only available for stochastic metafrontiers.
which	character. "meta" (default) returns the metafrontier (Stage 2) variance-covariance matrix; "group" returns a named list with one full variance-covariance matrix per group (from the inverse negative Hessian of the group MLE), with NULL entries for groups without a stored Hessian.
extraPar	logical. If TRUE and which = "meta", the full Stage 2 matrix is returned, including the rows and columns for the auxiliary parameters (raw MLE parameterisation, e.g. log_sigma_v); the default returns only the block for the frontier coefficients.
...	additional arguments (currently unused).

Value

A variance-covariance matrix (which = "meta"), a named list of matrices (which = "group"), or NULL if unavailable.

References

Murphy, K.M. and Topel, R.H. (1985). Estimation and inference in two-step econometric models. *Journal of Business & Economic Statistics*, 3(4), 370–379.

Examples

```
sim <- simulate_metafrontier(n_groups = 2, n_per_group = 50, seed = 42)
fit <- metafrontier(log_y ~ log_x1 + log_x2, data = sim$data,
                    group = "group", meta_type = "stochastic")
vcov(fit)

vcov(fit, correction = "murphy-topel")
```

Index