Package: metafrontier (via r-universe)

May 23, 2026

Type Package

Title Analysis of Metafrontier Models for Efficiency and Productivity

Version 0.2.2

Description Implements metafrontier production function models for estimating technical efficiencies and technology gaps for firms operating under different technologies. 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>. Additional features include panel SFA with time-varying inefficiency, bootstrap confidence intervals for technology gap ratios, latent class metafrontier estimation via the EM algorithm, Murphy-Topel corrected standard errors, 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, plm, ggplot2, parallel

VignetteBuilder knitr

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

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

Config/testthat/edition 3

Repository https://iik1.r-universe.dev

Date/Publication 2026-04-14 07:16:20 UTC

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

RemoteRef HEAD

RemoteSha 59f490de55f77a149647d603e553624c7e0677a3

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**, **frontier**, and **Benchmarking** 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.

Value

A list with components: coefficients, efficiency, 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)
```

---

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"), ...)
```

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.

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

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,
  method = c("dea", "sfa"),
  dist = c("hnormal", "tnormal", "exponential"),
  orientation = c("output", "input"),
  rts = c("crs", "vrs", "drs", "irs"),
  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.

`method`	character. `"dea"` (default) for DEA-based distance functions or `"sfa"` for SFA-based parametric distance functions.
`dist`	character. Distribution of the inefficiency term when `method = "sfa"`: `"hnormal"` (default), `"tnormal"`, or `"exponential"`.
`orientation`	character. `"output"` (default) or `"input"`.
`rts`	character. Returns to scale assumption: `"crs"` (default), `"vrs"`, `"drs"`, or `"irs"`.
`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

Computation uses DEA-based distance functions. 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.

Balanced panel assumption: Firms are matched across periods by position within each group. The data should contain a balanced panel (the same firms observed in every period) with consistent ordering. If group sizes differ across periods, only the first `min(n_s, n_t)` firms per group are paired and unmatched observations are silently dropped.

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)

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

orientation the orientation used

rts the returns to scale assumption

groups group labels

periods time periods

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

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
malm <- malmquist_meta(
  log_y ~ log_x1 + log_x2,
  data = panel_data,
  group = "group",
  time = "time"
)
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"),
  models = NULL,
  panel = NULL,
  panel_dist = c("bc92", "bc95"),
  type = c("radial", "directional"),
  direction = c("proportional", "output", "input"),
  control = list(),
  ...
)
```

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), or "irs" (increasing). Ignored when method = "sfa".
models	an optional named list of pre-fitted group-specific frontier models (objects from **sfaR**, **frontier**, or **Benchmarking**). If 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 or `"directional"` for directional distance functions.
`direction`	character. Direction vector for DDF: `"proportional"` (default), `"output"`, or `"input"`. 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).
`...`	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 (Battese, Rao, and O'Donnell, 2004) is estimated by solving a linear program that minimises the total envelope overshoot subject to the constraint that the metafrontier envelops all group frontiers. BRO (2004) originally proposed a constrained least-squares (QP) formulation; the LP yields the tightest envelope and is solved via **lpSolveAPI**, with a QP fallback via `constrOptim()` when the LP is infeasible. The stochastic metafrontier (Huang, Huang, and Liu, 2014) replaces this with a second-stage SFA, providing a distributional framework for inference on the TGR.

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

convergence convergence status

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. Uses a likelihood ratio test.

Usage

```
poolability_test(object, ...)
```

Arguments

`object`	a fitted `"metafrontier"` object with `method = "sfa"`.
`...`	additional arguments (currently unused).

Details

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).

Value

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

statistic the LR test statistic

parameter degrees of freedom

p.value p-value of the test

method description of the test

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)
```

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

Print a Metafrontier Object

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

Select Number of Latent Classes via BIC

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.

Details

Fits latent class metafrontier models for each value in n_classes_range and returns BIC values. The optimal number of classes minimises BIC.

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. 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
)
```

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, 0.5, 0.3)`.
`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.
`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.

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
```

---

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
)
```

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.

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

Examples

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

---

summary.metafrontier *Summary of a Metafrontier Model*

Description

Summary of a Metafrontier Model

Usage

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

Arguments

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

Value

An object of class "summary.metafrontier".

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.

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.

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.

---

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

Description

Variance-Covariance Matrix for Metafrontier Coefficients

Usage

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

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.
...	additional arguments (currently unused).

Value

A variance-covariance matrix, 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