

Package ‘sfa’

September 8, 2026

Version 1.2.0

Date 2026-09-07

Title Stochastic Frontier Analysis

Type Package

Maintainer David Bernstein <davebernstein1@gmail.com>

Description Provides a user-friendly framework for estimating a wide variety of cross-sectional and panel stochastic frontier models. Suitable for a broad range of applications, the implementation offers extensive flexibility in specification and estimation techniques.

Suggests knitr, rmarkdown, testthat, sn, np, lpSolve, sandwich, lmtest, texreg

Imports pso, numDeriv, gsl, plm, minqa, randtoolbox, mnormt, truncnorm, tmvtnorm, Formula, methods, pbapply, pracma, MASS, parallel, graphics, grDevices

Depends R (>= 4.0.0)

License GPL (>= 2)

Language en-US

URL <https://www.davidharrybernstein.com/software>,
<https://github.com/davidhbernstein/sfa>

BugReports <https://github.com/davidhbernstein/sfa/issues>

LazyLoad yes

NeedsCompilation no

VignetteBuilder knitr

Author David Bernstein [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-2267-5741>>),
Christopher Parmeter [aut],
Alexander Stead [aut]

Repository CRAN

Date/Publication 2026-09-07 23:40:02 UTC

Contents

sfa-package	3
calibrate_c	5
copsfm	7
cw_test	10
data_gen_cs	12
data_gen_p	16
density_weights	19
efficiency	20
efficiency_ci	22
endogeneity_test	23
esfm	25
FinnishElec	27
gof_test	28
hscore	30
hscore_select	31
Indian	33
inefficiency_test	34
influence_sfa	35
ivsfm	37
lcsfm	40
lcsfm_homogeneity	44
marginal_effects	47
meanefficiency	48
moment_range	50
npsfm	52
panel89	57
pcomposed	58
pcomposed_model	60
PL80_MVTN	61
predict.sfareg	63
print.sfareg	64
psfm	65
psfm_bootstrap	80
selsfm	85
sfareg-methods	89
sfa_diagnostics	91
sfm	93
sfma	104
simulation_se	106
skewness_test	108
spec_test	110
summary.sfareg	112
TIC	113
ttsfm	115
uhet_test	118
USUtilities	120

zsfm

121

Index

124

sfa-package	Stochastic Frontier Analysis
-------------	------------------------------

Description

Provides a user-friendly framework for estimating a wide variety of cross-sectional and panel stochastic frontier models. Suitable for a broad range of applications, the implementation offers extensive flexibility in specification and estimation techniques.

Details

The DESCRIPTION file:

Package: sfa

Version: 1.2.0

Date: 2026-09-07

Title: Stochastic Frontier Analysis

Type: Package

Authors@R: c(person("David", "Bernstein", email = "davebernstein1@gmail.com", role = c("aut", "cre"), comment =

Maintainer: David Bernstein <davebernstein1@gmail.com>

Description: Provides a user-friendly framework for estimating a wide variety of cross-sectional and panel stochastic

Suggests: knitr, rmarkdown, testthat, sn, np, lpSolve, sandwich, lmtest, texreg

Imports: pso, numDeriv, gsl, plm, minqa, randtoolbox, mnormt, truncnorm, tmvtnorm, Formula, methods, pbappt

Depends: R (>= 4.0.0)

License: GPL (>= 2)

Language: en-US

URL: <https://www.davidharrybernstein.com/software>, <https://github.com/davidhbernstein/sfa>

BugReports: <https://github.com/davidhbernstein/sfa/issues>

LazyLoad: yes

NeedsCompilation: no

VignetteBuilder: knitr

Author: David Bernstein [aut, cre] (ORCID: <<https://orcid.org/0000-0002-2267-5741>>), Christopher Parmeter [a

Index of help topics:

FinnishElec	FinnishElec
Indian	Indian
PL80_MVTN	Pitt and Lee (1981) Model III: the multivariate truncated normal panel likelihood
TIC	Choosing between non-nested stochastic frontier specifications
USUtilities	USUtilities
calibrate_c	Fixed robustness parameter by weight matching

<code>copsfm</code>	Stochastic Frontier with Dependence Between Noise and Inefficiency
<code>cw_test</code>	Chen and Wang (2012) centered-residuals moment test
<code>data_gen_cs</code>	Generate Cross-Sectional Data for Stochastic Frontier Analysis
<code>data_gen_p</code>	Generate Panel Data for Stochastic Frontier Analysis
<code>density_weights</code>	Density-power weights from a robust frontier fit
<code>efficiency</code>	Technical Efficiency Scores
<code>efficiency_ci</code>	Confidence Intervals for Individual Inefficiency and Efficiency
<code>endogeneity_test</code>	Were the regressors endogenous after all?
<code>esfm</code>	Stochastic frontier tolerating the wrong skewness
<code>gof_test</code>	Goodness-of-fit test for the assumed inefficiency distribution
<code>hscore</code>	Hyvarinen score for a Normal-Half-Normal frontier fit
<code>hscore_select</code>	Select the robustness tuning parameter by Hyvarinen score
<code>inefficiency_test</code>	Test whether there is any technical inefficiency
<code>influence_sfa</code>	Which observations are driving this frontier fit?
<code>ivsfm</code>	Stochastic Frontier Models with Endogenous Regressors
<code>lcsfm</code>	Latent Class Stochastic Frontier Models
<code>lcsfm_homogeneity</code>	Test a latent class frontier against homogeneity
<code>marginal_effects</code>	Marginal Effects of the Variance Determinants on Inefficiency
<code>meanefficiency</code>	Model-Implied Mean Efficiency
<code>moment_range</code>	What skewness and excess kurtosis can a pair of distributions produce?
<code>npsfm</code>	Nonparametric Stochastic Frontier Models
<code>panel89</code>	Panel89
<code>pcomposed</code>	The distribution of the composed error
<code>pcomposed_model</code>	Composed-error distribution function for any cross-sectional model
<code>predict.sfareg</code>	Predictions, fitted values and residuals for stochastic frontier fits
<code>print.sfareg</code>	Print method for "sfareg" objects
<code>psfm</code>	psfm
<code>psfm_bootstrap</code>	Parametric Bootstrap for 'psfm()' Panel Stochastic Frontier Models
<code>selsfm</code>	Stochastic Frontier Models with Sample

	Selection or Technology Choice
sfa-package	Stochastic Frontier Analysis
sfa_diagnostics	Optimizer Diagnostics for Stochastic Frontier Fits
sfareg-methods	Extractor methods for "sfareg" objects
sfm	sfm
sfma	Average over inefficiency distributions instead of choosing one
simulation_se	Simulation Standard Errors for Simulated-ML Fits
skewness_test	Formal tests for wrong skewness in the OLS residuals
spec_test	Is this pair of distributions defensible?
summary.sfareg	Summary method for "sfareg" objects
ttsfm	Two-Tier Stochastic Frontier Model
uhet_test	Test whether technical inefficiency depends on firm characteristics
zsfm	Zero-Inefficiency Stochastic Frontier Models

See Also

<https://www.davidharrybernstein.com/software>

<https://github.com/davidhbernstein/sfa>

Examples

```
## Simple application of the generalized true random effects estimator.
library(sfa)

data_trial <- data_gen_p(t=10,N=100, rand = 100,
                        sig_u = 1, sig_v = 0.3,
                        sig_r = .2, sig_h = .4,
                        cons = 0.5, beta1 = 0.5,
                        beta2 = 0.5)

psfm(formula = y_gtre ~ x1 + x2,
      model_name = "GTRE",
      data = data_trial,
      individual = "name",
      PSopt = FALSE)
```

Description

Chooses the robustness tuning parameter so that a residual a given number of scale units from the reference point retains a common share of its clean-data influence, comparably across the robust criteria. This is the transparent alternative to the data-driven selection of `hscore_select`.

Usage

```
calibrate_c(sigma_v, sigma_u, method = c("mlqe", "psi", "mdpd", "all"),
            target = 0.10, k = 3, range = c(1e-4, 0.6))
```

Arguments

<code>sigma_v, sigma_u</code>	Positive scalars at which to calibrate, typically the estimates from a preliminary fit or the values of a simulation design.
<code>method</code>	Robust criterion, or "all" for a named vector.
<code>target</code>	Share of clean-data influence the reference residual should retain. Default 0.10.
<code>k</code>	Reference residual in scale units. Default 3.
<code>range</code>	Search interval for the tuning parameter.

Details

The calibration matches the norm of the observationwise estimating-equation contribution in the intercept, $\log \sigma_v$ and $\log \sigma_u$ directions. Matching on the intercept direction alone would be vacuous: there the criteria share the density-power factor $f^{c-1} f'$, so it would force the same tuning value for all of them by construction. They differ through the Fisher-consistency correction, which enters the scale directions.

Note that $\sigma = (\sigma_v^2 + \sigma_u^2)^{1/2}$ is the scale parameter of the composed density, not the standard deviation of the composed error, which is $\{\sigma_v^2 + (1 - 2/\pi)\sigma_u^2\}^{1/2}$. The argument `k` is measured in the former.

The ratio is not monotone, and there can be two answers. For the Fisher-consistency-corrected criteria the influence ratio falls below the target and is then pulled back up again by the integral correction, so it crosses the target twice. At $\sigma_v = 0.30$, $\sigma_u = 0.60$ the Ψ /MDPD solutions are $c = 0.2387$ and $c = 0.5385$. The **larger** is returned: it is the calibration reported by Bernstein, Parmeter and Wright, and it lies on the far side of the dip, so the target down-weighting holds there rather than being passed through on the way. Both are in the "roots" attribute; the non-monotonicity is reported rather than resolved silently. MLqE has a single solution, $c = 0.2166$.

"psi" and "mdpd" always calibrate to the same value: the two objectives differ by the factor $(1 + c)$, which is constant in the parameters and cancels from the ratio being matched.

Value

A named numeric vector of calibrated tuning parameters, carrying an attribute "roots" that lists every solution found for each criterion.

References

Bernstein, D.H., Parmeter, C.F. and Wright, I.A. (2026). On Robust Estimation of the Stochastic Frontier Model. Working paper.

See Also

[hscore_select](#), [density_weights](#)

Examples

```
## the calibration used in Bernstein, Parmeter and Wright
calibrate_c(sigma_v = 0.30, sigma_u = 0.60, method = "all")
```

copsfm	<i>Stochastic Frontier with Dependence Between Noise and Inefficiency</i>
--------	---

Description

Fits a cross-sectional stochastic frontier in which the noise v and the inefficiency u are *dependent*, with the dependence carried by a copula. Every other model in the package assumes independence.

Usage

```
copsfm(formula, data,
        copula = c("gaussian", "fgm", "frank",
                    "clayton", "clayton90", "clayton180", "clayton270",
                    "gumbel", "gumbel90", "gumbel180", "gumbel270",
                    "joe", "joe90", "joe180", "joe270"), inefdec = TRUE,
        n_nodes = 128, maxit.bobyqa = 10000, maxit.psoptim = 1000,
        maxit.optim = 1000, start_val = FALSE, PSopt = FALSE,
        optHessian = TRUE, Method = "L-BFGS-B", verbose = FALSE,
        rand.psoptim = NULL)
```

Arguments

formula	Two-sided formula for the frontier, $y \sim x_1 + x_2$. A segment is an error: variance determinants are not yet supported alongside a copula.
data	A data.frame.
copula	The dependence family. See ‘Choosing a family’ below.

family	parameter	independence at	dependence it can express
"gaussian"	$\rho \in (-1, 1)$	0	both signs, no tail dependence
"fgm"	$\theta \in [-1, 1]$	0	both signs but weak : Spearman $\rho = \theta/3$
"frank"	$\theta \in \mathbb{R}$	0	both signs, full range, no tail dependence
"clayton"	$\theta > 0$	$\theta \rightarrow 0$	positive, lower -tail dependence
"gumbel"	$\theta \geq 1$	1	positive, upper -tail dependence

"joe"	$\theta \geq 1$	1	positive, heavier upper tail than Gumbel
Clayton, Gumbel and Joe carry <i>only positive</i> dependence, which is a real restriction: nothing rules out a negative association between noise and inefficiency, and against negatively dependent data those families can only report the independence boundary. The 90 and 270 suffixes are the rotations that reverse the sign, and 180 is the survival copula, which preserves it. So "clayton270" gives lower-tail dependence with a negative association.			
inefdec	TRUE for a production frontier, FALSE for cost.		
n_nodes	Gauss–Legendre nodes for the integral over u .		
maxit.bobyqa, maxit.psoptim, maxit.optim, start_val, PSopt, optHessian, Method, verbose, rand.psoptim	As in sfm .		

Details

The package’s taxonomy already writes the general joint density as

$$f_{V,U}(v, u) = f_V(v) f_U(u) c(F_V(v), F_U(u); \rho),$$

with every existing specification setting $c \equiv 1$. This function relaxes exactly that and nothing else: the marginals remain normal and half-normal. Composing over $v = \varepsilon + Su$,

$$f_\varepsilon(\varepsilon) = \int_0^\infty f_V(\varepsilon + Su) f_U(u) c(F_V(\varepsilon + Su), F_U(u)) du,$$

a one-dimensional integral evaluated by Gauss–Legendre quadrature on $u = t/(1 - t)$ rather than by simulation.

Why n_nodes defaults to 128. Measured against the closed-form normal/half-normal density at independence, the largest absolute error in $\log f$ is 1.3e-2 at 32 nodes, 2.5e-5 at 64, and 8.2e-14 at 128. Sixty-four looks adequate and is not: 1e-5 per observation is 0.02 in a log-likelihood over 2000 points, which is the scale at which competing modes are compared.

The dependence parameter needs a large sample. This is the thing to know before using it. The frontier slopes behave normally, but ρ does not. **Most of these families do not recover their own dependence parameter, and copsfm warns when you pick one.** A correct density does not imply an estimable parameter. Measured on 25 samples per family, generated from that family at $n = 400$ and refitted with it:

family	mean estimate (truth)	fits on the independence bound
"frank"	5.43 (5)	0%
"clayton"	2.24 (2)	0%
"gumbel"	1.29 (2)	36%
"joe"	1.52 (2)	40%
"clayton270"	0.23 (2)	56%
"gumbel190"	1.26 (2)	60%

On data generated from a Gumbel copula with Spearman $\rho = 0.685$ at $n = 2000$, *every* family – including the true one – returns the independence boundary, and their log-likelihoods differ by less

than 0.04. The likelihood is close to flat in the dependence parameter. This is a property of the model rather than of the implementation: each density is checked against the second mixed partial of its own CDF, and each sampler against the family's theoretical Spearman correlation. The remaining rotations were not measured, and warn that they were not, rather than implying either outcome.

Prefer "frank" or "clayton", and treat an estimate from any other family as exploratory.

Choosing a family, and a warning about doing so. The families differ in two things that matter: whether they admit negative dependence, and where they concentrate it. Gaussian and Frank spread association evenly and have no tail dependence; Clayton concentrates it in the lower tail and Gumbel and Joe in the upper; FGM can only express weak association at all.

Against that, weigh what the next paragraphs establish: the dependence parameter is estimated very imprecisely. On a Gaussian design at $n = 600$ its sampling standard deviation is 0.385 against a truth of 0.5. Offering fifteen families does not make that better, and it makes one thing worse – with fifteen candidates, some family will attain the highest likelihood on independent data by chance. On a sample generated with *independent* v and u , the fitted log-likelihoods across families span only about 0.8, and the best of them reports $\theta = 1.83$ for Gumbel. Treat a family comparison as descriptive, not as evidence of a dependence structure, and prefer a family chosen for a reason to one chosen by its likelihood.

On a Gaussian-copula design with true $\rho = 0.6$, $\sigma_U = 1$, $\sigma_V = 0.4$:

n	$\hat{\rho}$	$\hat{\sigma}_U$	$\hat{\sigma}_V$
2000	-0.247	0.571	0.275
8000	0.528	0.921	0.379
20000	0.571	0.971	0.389

It is consistent – the estimates converge on the truth – but at $n = 2000$ the sign can come out wrong. The cause is joint identification rather than a bad optimiser: with β and the scales held at their true values the likelihood peaks exactly at $\rho = 0.6$, so ρ is well identified *conditionally*; it is the trade-off against σ_U , σ_V and the intercept that defeats it in moderate samples, and the fitted point genuinely attains a higher likelihood than the truth.

Treat $\hat{\rho}$ as informative only in large samples, and compare against an independent fit (**sfm**) before reading anything into it.

And below a few hundred observations, the independent fit is often the better estimate – even when the dependence is real. Fitting the true copula gives a nearly unbiased but very noisy $\hat{\sigma}_U$; ignoring the copula gives a precise but biased one. RMSE of $\hat{\sigma}_U$ (truth 1.0) on Gaussian-copula data, 25 replications:

design	n	true copula	independence
$\rho = 0.5$	400	0.431	0.240
$\rho = 0.5$	1000	0.151	0.220
$\rho = -0.5$	400	0.251	0.196
$\rho = -0.5$	1000	0.175	0.203

The crossover sits between $n = 400$ and $n = 1000$. Below it the variance cost of estimating ρ jointly with σ_U exceeds the bias it removes, so plain **sfm** wins on RMSE; above it **copsfm** wins. The independence bias does not shrink with n (about 22% of σ_U , since it is misspecification rather than noise), while the copula fit's RMSE falls as it should.

Node counts, measured in paired runs. Forty datasets sent to every node count, so the differences are the quadrature and not the draw. Against the 128-node default, the mean difference in $\hat{\rho}$ is +0.0002 at 256 nodes ($t = 1.37$) and +0.0046 at 64 ($t = 1.47$) – neither distinguishable – but **-0.0783 at 32** ($t = -2.79$, $p = 0.008$), with per-dataset differences as large as 0.52. So the quadrature has converged by 64, and 32 must not be used as a fast option: it is biased, not merely noisier. For scale, $\hat{\rho}$'s own sampling standard deviation is 0.385 at $n = 600$, which dwarfs every node-count effect here.

Families. Only densities that could be verified are offered. Each is checked in the tests two ways: it integrates to 1 over the unit square, and it returns exactly 1 at the independence parameter. Frank, Clayton and Gumbel are deliberately absent rather than transcribed without a source to check against.

Value

An object of class "sfareg", with `copula` and `copula_par` alongside the usual components. `jlm` is $E[u \mid \varepsilon]$ computed by the same quadrature the likelihood used.

References

Smith, M. D. (2008). Stochastic frontier models with dependent error components. *The Econometrics Journal*, **11**(1), 172–192.

See Also

[sfm](#) for the independent case.

Examples

```
set.seed(4)
## n_nodes = 64 rather than the default 128: the quadrature is converged by 64
## (see Details), and this is the slowest entry point in the package.
n <- 600
x1 <- rnorm(n); x2 <- rnorm(n)
z1 <- rnorm(n); z2 <- 0.6 * z1 + sqrt(1 - 0.6^2) * rnorm(n)
y <- 0.5 + 0.8 * x1 - 0.4 * x2 + 0.4 * z1 - qnorm((1 + pnorm(z2)) / 2)
dat <- data.frame(y = y, x1 = x1, x2 = x2)

fit <- copsfm(y ~ x1 + x2, data = dat, n_nodes = 64)
fit$out
```

cw_test

Chen and Wang (2012) centered-residuals moment test

Description

Tests the distributional assumption on the composed error using the characteristic function of the centred residuals, without ever evaluating the composed-error density or distribution function.

Usage

```

cw_test(x, model_name = NULL, tau = 1, type = c("cosine", "sine"),
        data = NULL, n_nodes = 256L)

```

Arguments

x	An "sfareg" fit, or a numeric vector of regression residuals – ordinary least squares residuals are enough.
model_name	The distributional pair under the null. Required when x is a numeric vector; taken from the fit otherwise. One of "NHN", "NE", "NR", "NU", "NGE".
tau	Frequency at which the characteristic function is compared. Chen and Wang suggest a value around 1, and a <i>single</i> frequency: combining several oversizes the test (see Details).
type	"cosine" (recommended) or "sine".
data	The data the model was fitted to, when x is a fit.
n_nodes	Quadrature nodes for the theoretical moments.

Details

Why this test rather than [gof_test](#). It never touches the composed-error PDF or CDF. It compares the empirical characteristic function of the centred residuals with its theoretical value, and that follows from the characteristic functions of the two components *separately*, which are in closed form for many pairs whose convolution is not – the normal-gamma model being the standard example of a pair with no closed-form composed density.

Centring is what makes it work. A frontier's intercept is not identified separately from $E[u]$, so the least-squares intercept estimates $\alpha - E[u]$ rather than α . Centring the residuals cancels that inconsistent intercept outright, which is why the test is valid off an ordinary regression and does not require maximum likelihood. The variance then has to correct for two nuisances – estimating the scale parameters, and the centring itself. Omitting the second understates the variance and oversizes the test.

Use one frequency, not several. Measured here over 500 replications at a nominal 5%, half-normal data:

test	n	$\tau = 1$	$\tau = 1.5$	$\tau = 2$
cosine	500	0.070	0.054	0.050
cosine	2000	0.046	0.044	0.050
sine	500	0.092	0.064	0.042
sine	2000	0.068	0.062	0.064

Against exponential inefficiency the cosine test's power at $\tau = 1$ is 0.231 at $n = 500$ and 0.952 at $n = 2000$, against 0.108 and 0.706 for the sine test. Both of Chen and Wang's conclusions reproduce: the cosine test is the less sensitive to the choice of τ , and the more powerful. Passing several frequencies at once gives rejection rates of 0.09 to 0.15 at a nominal 5%, because the cosine moments at nearby frequencies are close to collinear and the covariance matrix is then near-singular; the authors report the same for their combined τ .

The moment estimator solves the second- and third-moment equations, which for "NHN" and "NE" is the closed form of the paper's equation (32) and is the same inversion `sfm(estimator = "cols")` uses. A wrongly signed third moment leaves it with no admissible solution, and that is an error rather than a number: see [skewness_test](#).

Value

An object of class "cw_test": the statistic D, its degrees of freedom and p-value, the moment estimates of σ_v and σ_u , and the frequencies used.

References

Chen, Y.-T. and Wang, H.-J. (2012). Centered-residuals-based moment estimator and test for stochastic frontier models. *Econometric Reviews* **31**(6), 625–653.

See Also

[gof_test](#), [spec_test](#), [skewness_test](#)

Examples

```
set.seed(4)
n <- 500
X <- matrix(rnorm(n * 3), n, 3)
y <- X %*% c(1, 1, 1) + rnorm(n, 0, 1) - abs(rnorm(n, 0, 1))
e <- residuals(lm(y ~ X))

cw_test(e, "NHN")           # true pair
cw_test(e, "NE")           # wrong inefficiency distribution
```

data_gen_cs

Generate Cross-Sectional Data for Stochastic Frontier Analysis

Description

`data_gen_cs` generates simulated cross-sectional data based on the stochastic frontier model, allowing for different distributional assumptions for the one-sided technical inefficiency error term (u) and the two-sided idiosyncratic error term (v). The model has the general form: $Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + v - u$ where $u \geq 0$ and represents inefficiency. All variants are produced so that the user can select those that they want.

Usage

```
data_gen_cs(N, rand, sig_u, sig_v, cons, beta1, beta2, a, mu, sig_w = sig_u,
            shape_g = 2, m_nak = 1, mu_ln = -0.5, k_w = 1.5, lam_tsl = 1.5)
```

Arguments

N	A single integer specifying the number of observations (cross-sectional units).
rand	A single integer to set the seed for the random number generator, ensuring reproducibility.
sig_u	The standard deviation parameter (σ_u) for the base distribution of the one-sided error term u .
sig_v	The standard deviation parameter (σ_v) for the base distribution of the two-sided error term v .
cons	The value of the constant term (intercept) in the model.
beta1	The coefficient for the x_1 variable.
beta2	The coefficient for the x_2 variable.
a	The degrees of freedom parameter for the t half-t distribution (u_t and v_t , respectively). Requires the <code>rt</code> function.
mu	The mean parameter (μ) for the normal truncated normal distribution (u_{tn}). Requires the <code>rtruncnorm</code> function.
sig_w	The standard deviation/scale parameter (σ_w) for the second one-sided error component w , used to generate two-tier model columns (see ttsfm). Defaults to <code>sig_u</code> .
shape_g	Shape of the gamma inefficiency draw behind <code>y_pcs_g</code> , which targets <code>sfm(model_name = "NG")</code> . The scale is set to <code>sig_u/shape_g</code> , so $E[u] = \sigma_u$ regardless of the shape chosen. <code>sfm()</code> 's NG likelihood reports the shape as <code>mu</code> and the scale as <code>sig_u</code> , so the values to recover are <code>shape_g</code> and <code>sig_u/shape_g</code> .
m_nak	Nakagami shape m behind <code>y_pcs_nak</code> , which targets <code>sfm(model_name = "NNAK")</code> . The spread is $\Omega = \sigma_u^2$, so σ_u is the root-mean-square of u . The default 1 is the Rayleigh case; 0.5 would collapse the Nakagami onto the half-normal and so duplicate <code>y_pcs</code> .
mu_ln	Log-scale mean of the lognormal inefficiency draw behind <code>y_pcs_ln</code> , which targets <code>sfm(model_name = "NLN")</code> . The log-scale standard deviation is <code>sig_u</code> . The default -0.5 with <code>sig_u = 1</code> gives $E[u] = 1$.
k_w	Weibull shape behind <code>y_pcs_wb</code> , which targets <code>sfm(model_name = "NW")</code> . The scale is <code>sig_u</code> .
lam_tsl	Skew parameter λ behind <code>y_pcs_tsl</code> , which targets <code>sfm(model_name = "TSL")</code> . The scale is <code>sig_u</code> ; $\lambda \rightarrow 0$ recovers the exponential case.

Details

The function simulates two explanatory variables, x_1 and x_2 , as transformations of uniform random variables.

The function generates several different frontier models by combining various distributions for u and v :

- **u Distributions (Inefficiency):** Half-Normal (HN), Truncated Normal (TN), Half-T (HT), Half-Cauchy (HC), Exponential (E), Half-Uniform (HU).
- **v Distributions (Idiosyncratic):** Normal (N), t, Cauchy (C).

****Specific Model Outputs (y_pcs variants):****

- **y_pcs**: Normal-Half Normal (N-HN): $v \sim N(0, \sigma_v^2)$, $u \sim |N(0, \sigma_u^2)|$.
- **y_pcs_z**: N-HN with Heteroskedastic σ_u : $\sigma_{u,i} = \exp(0.9 + 0.6Z_i)$, where Z is a uniform variable.
- **y_pcs_t**: T-Half T (T-HT): $v \sim T(\text{df} = a) \cdot \sigma_v$, $u \sim |T(\text{df} = a)| \cdot \sigma_u$.
- **y_pcs_st**: the column that actually matches **sfm**'s THT. **y_pcs_t** above draws its two components as two *independent* **rt()** variates, which shares the degrees of freedom but not the mixing variable, so the composed error is not skew-t and THT cannot recover it. **y_pcs_st** uses the single common $\lambda \sim \Gamma(a/2, a/2)$ of Tancredi (2002), i.e. $y = h + (\epsilon - z)/\sqrt{\lambda}$. **y_pcs_t** is retained unchanged only for backward compatibility.
- **y_pcs_thn**: Student's t-half normal, matching **sfm**'s tHN: $v \sim T(\text{df} = a) \cdot \sigma_v$ and $u \sim |N(0, \sigma_u^2)|$, drawn *independently*. Heavy-tailed noise with a conventional one-sided term – distinct from both **y_pcs_t** and **y_pcs_st**, in which the inefficiency is heavy-tailed too.
- **y_pcs_tn**: Normal-Truncated Normal (N-TN): $v \sim N(0, \sigma_v^2)$, $u \sim TN(\mu, \sigma_u^2)$ on $[0, \infty)$.
- **y_pcs_e**: Normal-Exponential (N-E): $v \sim N(0, \sigma_v^2)$, $u \sim \text{Exp}(\phi)$, where $\phi = 1/\sigma_u$.
- **y_pcs_c**: Cauchy-Half Cauchy (C-HC): $v \sim \text{Cauchy}(0, \sigma_v)$, $u \sim |\text{Cauchy}(0, \sigma_u)|$.
- **y_pcs_u**: Normal-Half Uniform (N-HU): $v \sim N(0, \sigma_v^2)$, $u \sim U(0, \sigma_u)$.
- **y_pcs_w**: Normal + Cauchy - Half Normal: $v \sim N(0, \sigma_v^2) + \text{Cauchy}(0, \sigma_v)$, $u \sim |N(0, \sigma_u^2)|$. This introduces a composite v term.

****Note:**** The **rtruncnorm** function is required for **y_pcs_tn** and loads with the package. In isolation it could be loaded by using **library(truncnorm)**.

Value

A data frame containing N observations with the following columns:

name	Individual identifier (simply 1 to N).
cons	The constant term value.
x1	Simulated explanatory variable x_1 .
x2	Simulated explanatory variable x_2 .
u, uz, u_t, u_c, u_e, u_u, u_tn, u_st, u_thn	The simulated one-sided error terms under different distributions.
u_r	Rayleigh one-sided error term, for y_pcs_r . Drawn on the convention sfm 's "NR" uses, $E[u^2] = \sigma_u^2$, so the Rayleigh scale is $\sigma_u/\sqrt{2}$, $E[u] = \sigma_u\sqrt{\pi}/2$ and $\text{Var}(u) = (1 - \pi/4)\sigma_u^2$.
v, v_t, v_c, v_st, v_thn	The simulated two-sided error terms under different distributions.
lam_st	The common Gamma($a/2$, $a/2$) scale-mixing variable behind y_pcs_st ; both of that column's components are divided by its square root, which is what makes the composed error skew-t.
y_pcs, y_pcs_t, y_pcs_st, y_pcs_thn, y_pcs_e, y_pcs_ez, y_pcs_c, y_pcs_u, y_pcs_z, y_pcs_w, y_pcs_tn	The dependent variable Y under the corresponding SFA model distributions.

y_pcs_r	Normal-Rayleigh dependent variable, for sfm 's "NR". Use this column, not y_pcs: the normal-Rayleigh and normal-half-normal composed errors are different families, not reparameterizations of one another, and "NR" cannot recover y_pcs's parameters. True values are $(\sigma_v, \sigma_u) = (\text{sig}_v, \text{sig}_u)$.
z	The auxiliary variable used for heteroskedasticity in y_pcs_z, y_pcs_ez, y_tthn_z, and y_zisf_z.
uz_e	Exponential one-sided error term with heteroskedastic scale, for y_pcs_ez.
w_tt, w_tt_hn, wz_hn	The second one-sided error component (w) for the two-tier columns below, under exponential, homoskedastic half-normal, and heteroskedastic half-normal distributions respectively.
zp	A second auxiliary variable used for heteroskedasticity of σ_w in y_tthn_z.
y_ttne	Homoskedastic two-tier dependent variable ($v + w - u$, both one-sided components exponential), for ttsfm 's "TTNE"/"TTNLS".
y_tthn	Homoskedastic two-tier dependent variable (both one-sided components half-normal), for ttsfm 's "TTHN" with no pipes.
y_tthn_z	Heteroskedastic two-tier dependent variable (σ_u a function of z , σ_w a function of zp), for ttsfm 's "TTHN" with formula $\sim x1 + x2 z zp$.
eff_ind, eff_ind_z	Indicator (1 = efficient, u forced to 0) used to build the zero-inefficiency columns below.
probab_z_true	The true heteroskedastic "efficient regime" probability used to draw eff_ind_z.
y_zisf	Zero-inefficiency dependent variable with constant efficient-regime probability, for zsfm 's "ZISF".
y_zisf_z	Zero-inefficiency dependent variable with z -dependent efficient-regime probability, for zsfm 's "ZISF_Z".
u_g, y_pcs_g	Gamma one-sided error term and the corresponding dependent variable, for sfm 's "NG". Shape shape_g, scale sig_u/shape_g, so $E[u] = \sigma_u$ whatever shape is chosen.
u_nak, y_pcs_nak	Nakagami one-sided error term and dependent variable, for sfm 's "NNAK". Shape m_nak, spread $\Omega = \sigma_u^2$. The default m_nak = 1 is the Rayleigh case, so this column coincides in distribution with y_pcs_r at the default.
u_ge, y_pcs_ge	Generalized-exponential one-sided error term and dependent variable, for sfm 's "NGE", drawn as $u = -\sigma_u \log(1 - \sqrt{U})$ with $U \sim \text{Unif}(0, 1)$.
u_ln, y_pcs_ln	Lognormal one-sided error term and dependent variable, for sfm 's "NLN". Log-scale mean mu_ln and log-scale standard deviation sig_u.
u_w, y_pcs_wb	Weibull one-sided error term and dependent variable, for sfm 's "NW". Shape k_w, scale sig_u.
con	A constant column set to 1, potentially for use in estimation.

Author(s)

David H. Bernstein

See Also

[data_gen_p](#) for the panel generator, and [sfm](#), [zsfm](#) and [ttsfm](#) for the estimators these columns are built for.

[rnorm](#), [runif](#), [rt](#), [rexp](#), [rcauchy](#), [rtruncnorm](#) (if available).

Examples

```
# Generate 100 observations of SFA data
data_sfa <- data_gen_cs(
  N      = 100,
  rand   = 123,
  sig_u  = 0.5,
  sig_v  = 0.2,
  cons   = 5,
  beta1  = 1.5,
  beta2  = 2.0,
  a      = 5, # degrees of freedom for T/Half-T
  mu     = 0.1 # mean for Truncated Normal
)

# Display the first few rows of the generated data
head(data_sfa)

# Example of a Normal-Half Normal SFA model data
summary(data_sfa$y_pcs)
plot(density(data_sfa$y_pcs))
```

data_gen_p

Generate Panel Data for Stochastic Frontier Analysis

Description

`data_gen_p` generates simulated panel data for estimating various panel stochastic frontier models, including the Generalized True Random Effects (GTRE), True Random Effects (TRE), Pooled Cross-Section (PCS), and True Fixed Effects (TFE) models. The function returns the data as a `pdata.frame`. All variants are produced so that the user can select those that they want.

Usage

```
data_gen_p(t, N, rand, sig_u, sig_v, sig_r, sig_h, cons, tau = 0.5,
           mu = 0, beta1, beta2, eta = 0.1, b_k90 = 0.05, c_k90 = 0.01,
           d_k90 = 0.05, e_k90 = -0.005, rho_mvtn = 0.5)
```

Arguments

<code>t</code>	The number of time periods.
<code>N</code>	The number of individuals.
<code>rand</code>	A seed for the random number generator to ensure reproducibility.

sig_u	The standard deviation (σ_u) for the one-sided error component (u_{it}).
sig_v	The standard deviation (σ_v) for the two-sided error component (v_{it}).
sig_r	The standard deviation (σ_r) for the two-sided individual effect (r_i).
sig_h	The standard deviation (σ_h) for the one-sided individual effect (h_i).
cons	The constant term (β_0) for the frontier models.
tau	The dependence parameter (τ) used for the y_tfe (TFE) model formulation, default is 0.5. See Chen, Schmidt, and Wang (2014, Journal of Econometrics).
mu	The mean parameter (μ) used for the Truncated-Normal (TN) component of the y_fd model with default set to 0. See Wang and Ho (2010, Journal of Econometrics).
beta1	The coefficient for the x1 variable (β_1).
beta2	The coefficient for the x2 variable (β_2).
eta	The decay-rate parameter (η) for the time-varying inefficiency component u_bc92, following Battese and Coelli (1992): $u_{it} = u_i \exp(-\eta(T_i - t))$. Default is 0.1. See Battese and Coelli (1992, Journal of Productivity Analysis).
b_k90, c_k90	Coefficients of the Kumbhakar (1990) inefficiency path $B_{it} = (1 + \exp(bt + ct^2))^{-1}$ behind y_k1990, which targets psfm(model_name = "K1990"). The defaults give a path running from 0.485 to 0.182 over ten periods. Note that b and c are weakly identified: quite different pairs produce near-identical fitted paths.
rho_mvtn	Within-firm correlation of inefficiency across periods for the y_pl_mvtn column, which targets psfm(model_name = "PL80_MVTN"). The inefficiency vector is drawn from $N(0, \sigma_u^2[(1 - \rho)I + \rho \mathbf{1}\mathbf{1}'])$ truncated to the negative orthant, so 0 gives inefficiency independent over time and values near 1 approach the time-invariant "PL80" case. Default 0.5. Note the realized correlation of the drawn column is ATTENUATED relative to rho by the truncation (about 0.29 at rho = 0.5, $T = 4$) – rho parameterizes the underlying normal, not the truncated draw.
d_k90, e_k90	Coefficients of the modified Kumbhakar path $B_{it} = 1 + d(t - T_i) + e(t - T_i)^2$ behind y_k1990m, which targets psfm(model_name = "K1990modified"). The defaults give a path running from 0.145 to 1 over ten periods.

Details

A `pdata.frame` object with $N \times t$ observations, containing the following columns:

- name Individual identifier.
- year Time period identifier.
- cons The constant term used in the data generation.
- x1, x2 Explanatory variables generated from a log-uniform distribution.
- x1_w, x2_w Explanatory variables with dependence parameter τ and linkage with r_i , used for the TFE model.
- u, v, r, h The generated error and individual effect components.
- y_gtre, y_tre, y_pcs, y_tfe Output variables for the Production Frontier models, including the constant.

- `y_gtre_nc`, `y_tre_nc`, `y_pcs_nc` Output variables for the Production Frontier models, **excluding** the constant.
- `c_gtre`, `c_tre`, `c_pcs`, `c_tfe` Output variables for the Cost Frontier models, including the constant.
- `c_gtre_nc`, `c_tre_nc`, `c_pcs_nc` Output variables for the Cost Frontier models, **excluding** the constant.
- `y_fd` Output variable for the first difference model (see Wang and Ho, 2010).
- `x_fd` Explanatory variable for the `y_fd` model.
- `u_fd_star`, `z_fd`, `r_fd`, `u_fd` Components used to generate `y_fd`.
- `u_gtre`, `z_gtre`, `zp_gtre`, `y_gtre_z`, `y_gtre_zz`, `y_tre_z` Variables for models with heteroskedastic inefficiency ($\sigma_{u,i} = \exp(0.33 + 0.61Z_i)$). Variables for models with heteroskedastic persistent inefficiency ($\sigma_{h,i} = \exp(0.25 + 0.21Z_i)$).
- `u_inv`, `y_ssfe` Time-invariant firm-level inefficiency ($u_i \sim |N(0, \sigma_u^2)|$, constant within individual across t) and the corresponding output variable, for `psfm`'s "SSFE" and "PL80".
- `u_bc92`, `y_bc92` Time-varying inefficiency following a Battese-Coelli (1992) decay structure ($u_{it} = u_i \exp(-\eta(T_i - t))$) and the corresponding output variable, for `psfm`'s "BC92".
- `B_k90`, `u_k90`, `y_k1990` The Kumbhakar (1990) inefficiency path $B_{it} = (1 + \exp(bt + ct^2))^{-1}$ (controlled by `b_k90`, `c_k90`), the resulting inefficiency $u_{it} = B_{it}u_i$, and the corresponding output variable, for `psfm`'s "K1990".
- `B_k90m`, `u_k90m`, `y_k1990m` The modified Kumbhakar path $B_{it} = 1 + d(t - T_i) + e(t - T_i)^2$ (controlled by `d_k90`, `e_k90`), the resulting inefficiency, and the corresponding output variable, for `psfm`'s "K1990modified".
- `h_z` The heteroskedastic persistent inefficiency h_i expanded to one value per observation (constant within individual), used to build `y_gtre_zz`.

The data is generated based on standard Stochastic Frontier Analysis (SFA) formulations, primarily for a ****Production Frontier**** where the one-sided error component u_{it} is subtracted:

- `y_gtre`: GTRE model: $y_{it} = \beta_0 + \beta_1 x_{1,it} + \beta_2 x_{2,it} + r_i - h_i + v_{it} - u_{it}$
- `y_tre`: TRE model: $y_{it} = \beta_0 + \beta_1 x_{1,it} + \beta_2 x_{2,it} + r_i + v_{it} - u_{it}$
- `y_pcs`: PCS model: $y_{it} = \beta_0 + \beta_1 x_{1,it} + \beta_2 x_{2,it} + v_{it} - u_{it}$
- `y_tfe`: TFE model: $y_{it} = \beta_1 x_{1,it}^w + \beta_2 x_{2,it}^w + r_i + v_{it} - u_{it}$
- `y_gtre_z`: GTRE with Heteroskedastic u_{it} : $\sigma_{u,i} = \exp(0.33 + 0.61Z_i)$.
- `y_gtre_zz`: GTRE with Heteroskedastic u_{it} : $\sigma_{u,i} = \exp(0.33 + 0.61Z_i)$ and $\sigma_{h,i} = \exp(0.25 + 0.21Z_i)$.

For ****Cost Frontier**** models, the one-sided error component u_{it} is added (e.g., `c_gtre`).

The error terms are generated as:

- $r_i \sim N(0, \sigma_r^2)$ (individual two-sided effect)
- $h_i \sim |N(0, \sigma_h^2)|$ (individual one-sided effect)
- $v_{it} \sim N(0, \sigma_v^2)$ (two-sided noise)
- $u_{it} \sim |N(0, \sigma_u^2)|$ (one-sided inefficiency)

The First-Difference estimation model (`y_fd`) uses a variation where $r_{i,fd} \sim U(0, 1)$ and $u_{it,fd}$ is generated using a heteroskedastic truncated-normal structure, reflecting an alternative model type.

Value

A `pdata.frame` object containing $N \times t$ observations suitable for Stochastic Frontier Analysis (SFA).

Author(s)

David H. Bernstein

References

- Chen, Y., Schmidt, P., & Wang, H. (2014). Consistent estimation of the fixed effects stochastic frontier model. *Journal of Econometrics*, 181(2), 65-76.
- Filippini, M., & Greene, W. H. (2016). Persistent and transient productive inefficiency: a maximum simulated likelihood approach. *Journal of Productivity Analysis*, 45, 187-196.
- Wang, H., & Ho, C. M. (2010). Estimating fixed-effect panel stochastic frontier models by model transformation. *Journal of Econometrics*, 157(2), 286-296.

See Also

`data_gen_cs` for the cross-sectional generator, and `psfm` for the estimators these columns are built for.

Examples

```
library(sfa)
# Generate a dataset
data_trial <- data_gen_p(t=10, N=100, rand = 100,
                        sig_u = 1, sig_v = 0.3,
                        sig_r = .2, sig_h = .4,
                        cons = 0.5, tau = 0.5,
                        mu= 0.5, beta1 = 0.5,
                        beta2 = 0.5)

# See the first few rows
head(data_trial)
```

density_weights	<i>Density-power weights from a robust frontier fit</i>
-----------------	---

Description

Returns the weight each observation receives in the estimating equation of a maximum L_q -likelihood or minimum density-power divergence fit. The weight is the fitted density raised to the power c , normalised so the largest is one.

Usage

```
density_weights(object, sigma_v = NULL, sigma_u = NULL, c = NULL,
               normalize = TRUE)
```

Arguments

object	A fitted frontier model, or a numeric vector of composed residuals.
sigma_v, sigma_u	Scale parameters. Required when object is a numeric vector; taken from the fit otherwise.
c	Robustness tuning parameter. Taken from the fit when available.
normalize	Logical; scale so the maximum weight is one. Default TRUE.

Value

A numeric vector of weights in $(0, 1]$, one per observation.

What this diagnostic can and cannot find

The weight is a function of the fitted density at the observation, so it responds to observations lying far from the fitted surface. It has no purchase on an observation that is wrong in a *regressor* and, because the fitted surface bends toward it, ends up close to that surface. A mis-recorded input can therefore be highly influential and still receive a weight near one. This is not a defect of the implementation: the bounded influence of these estimators is bounded with respect to *response* contamination, conditional on a fixed or bounded design. Pair this with a leverage diagnostic, and with case deletion, before concluding that an observation is or is not driving a result.

References

Bernstein, D.H., Parmeter, C.F. and Wright, I.A. (2026). On Robust Estimation of the Stochastic Frontier Model. Working paper.

See Also

[hscore_select](#), [calibrate_c](#)

Examples

```
set.seed(1)
e <- rnorm(200, 0, 0.3) - abs(rnorm(200, 0, 0.6))
w <- density_weights(e, sigma_v = 0.3, sigma_u = 0.6, c = 0.217)
summary(w)
head(order(w), 10) # the observations the estimator discounts most
```

efficiency	<i>Technical Efficiency Scores</i>
------------	------------------------------------

Description

Extracts observation-specific technical efficiency from a fitted frontier, choosing among the three usual point predictors and between the two scales the dependent variable may be on.

Usage

```
efficiency(object, type = c("bc", "jlms", "mode"), logDepVar = TRUE,
           newdata = NULL)
```

Arguments

<code>object</code>	An "sfareg" fit.
<code>type</code>	Which predictor of u_i to use. "bc" (the default) is Battese–Coelli, $E[\exp(-u_i) \mid \varepsilon_i]$; "jlms" is Jondrow <i>et al.</i> , $\exp(-E[u_i \mid \varepsilon_i])$; "mode" uses the mode of the posterior of u_i .
<code>logDepVar</code>	TRUE (the default) if the dependent variable is on a log scale, so that efficiency is $\exp(-u_i)$; FALSE if it is on a level scale, giving $1 - u_i/f(x_i)$.
<code>newdata</code>	Data to rebuild the fitted frontier from, needed only when <code>logDepVar = FALSE</code> . Supply it when the data used for the fit can no longer be recovered from the stored call.

Details

Which predictor. The three differ in what they report about the same posterior, not in how well they are estimated. $E[\exp(-u)] \geq \exp(-E[u])$ by Jensen's inequality, so "bc" is never below "jlms"; the gap widens as the posterior spreads out. "mode" is the only one that can equal exactly 1, and it does so for every observation whose posterior mean is negative – typically much of the efficient tail. That is a property of the predictor rather than a defect: where the posterior mean of u is negative the single most likely value really is the boundary. It is available only for the models whose posterior is a truncated normal ("NHN", "NHN_Z", "NE", "NTN").

Which scale. The package elsewhere assumes throughout that the dependent variable is logged, which is the usual case and makes u a proportional shortfall. `logDepVar = FALSE` instead forms $(f(x_i) - u_i)/f(x_i)$. That is a ratio of two estimated quantities and is less well behaved than the log version: it is undefined where the fitted frontier is zero and negative where u exceeds it, both of which happen near zero output. A warning is issued if any score comes back negative or non-finite, since silence there would be the real failure.

Value

A numeric vector of efficiency scores, one per observation used in the fit.

See Also

[efficiency_ci](#) for interval estimates of the same quantity.

Examples

```
set.seed(2)
n <- 400
x1 <- rnorm(n); x2 <- rnorm(n)
y <- 6 + 0.5 * x1 + 0.5 * x2 + rnorm(n, 0, 0.3) - abs(rnorm(n, 0, 0.8))
d <- data.frame(y = y, x1 = x1, x2 = x2)
```

```
fit <- sfm(y ~ x1 + x2, model_name = "NHN", data = d)

summary(efficiency(fit))                # Battese-Coelli, log scale
summary(efficiency(fit, type = "jlms")) # never above the above
summary(efficiency(fit, type = "mode"))  # reaches exactly 1
summary(efficiency(fit, logDepVar = FALSE, newdata = d))
```

efficiency_ci

Confidence Intervals for Individual Inefficiency and Efficiency

Description

Horrace and Schmidt (1996) intervals for the inefficiency term u_i of a fitted stochastic frontier model, and the implied intervals for technical efficiency $\exp(-u_i)$.

Usage

```
efficiency_ci(object, level = 0.95, type = c("both", "u", "te"))
```

Arguments

object	an object of class "sfareg", as returned by <code>sfm</code> .
level	coverage of the interval, a single number strictly between 0 and 1. Defaults to 0.95.
type	which intervals to return: "both" (the default), "u" for inefficiency only, or "te" for technical efficiency only.

Details

Conditional on the fitted parameters, $u_i \mid e_i$ is normal with mean `mu_star` and standard deviation `sigma_star`, truncated below at zero – the same posterior the Jondrow et al. (1982) and Battese and Coelli (1988) point predictors average over. Inverting it at $(1 - \alpha)/2$ and $1 - (1 - \alpha)/2$ gives the bounds in closed form, so the interval costs no estimation beyond the fit itself. The efficiency bounds follow by monotonicity, with the endpoints swapped, since $\exp(-u)$ is decreasing in u .

These intervals condition on the estimated parameters. They describe where u_i sits given e_i and a *known* frontier, and make no allowance for sampling error in the slopes or the variance parameters. Horrace and Schmidt are explicit about this, and it is why the intervals do not narrow as the sample grows: they measure the irreducible difficulty of splitting a single residual into noise and inefficiency, not estimation uncertainty. Read a wide interval as a warning against reading much into that unit's rank.

Available for the models whose posterior really is a truncated normal: "NHN", "NHN_Z", "NE" and "NTN". Every other `model_name` has a posterior of a different shape, for which these formulas do not hold; the function reports that rather than returning a misleading number.

Value

A data frame with one row per observation, carrying `u_lower`, `u_hat`, `u_upper` and/or `te_lower`, `te_hat`, `te_upper` according to type, and a "level" attribute. `u_hat` is the Jondrow et al. (1982) posterior mean and `te_hat` the Battese and Coelli (1988) score – exactly the values the fit already reports, taken from the fitted object rather than recomputed.

References

- Horrace, W. C. and Schmidt, P. (1996). Confidence statements for efficiency estimates from stochastic frontier models. *Journal of Productivity Analysis*, **7**, 257–282.
- Jondrow, J., Lovell, C. A. K., Materov, I. S. and Schmidt, P. (1982). On the estimation of technical inefficiency in the stochastic frontier production function model. *Journal of Econometrics*, **19**, 233–238.
- Battese, G. E. and Coelli, T. J. (1988). Prediction of firm-level technical efficiencies with a generalized frontier production function and panel data. *Journal of Econometrics*, **38**, 387–399.

See Also

[sfm](#), [sfa_diagnostics](#)

Examples

```
dat <- data_gen_cs(N = 200, rand = 1, sig_u = 1, sig_v = 0.3,
                  cons = 0.5, beta1 = 0.5, beta2 = 0.5, a = 1, mu = 0.5)
fit <- sfm(y_pcs ~ x1 + x2, data = dat, model_name = "NHN")

ci <- efficiency_ci(fit, level = 0.90)
head(ci)

## How much of an efficiency ranking is real? Compare the spread of the point
## predictions against the width of a single unit's interval.
diff(range(ci$te_hat))
median(ci$te_upper - ci$te_lower)
```

endogeneity_test

Were the regressors endogenous after all?

Description

A Wald test of $H_0 : \rho = 0$ on an `ivsfm` fit – the hypothesis that the noise v is uncorrelated with the reduced-form errors of the endogenous regressors, so that the endogeneity correction was not needed and a plain `sfm` fit would have been consistent. This is the Durbin–Wu–Hausman idea carried into the stochastic frontier, and is Equation (25) of Hou, Ramalho and Roseta-Palma (2025).

Usage

```
endogeneity_test(object, level = 0.05)
```

Arguments

object	A fit from <code>ivsfrm</code> with <code>model_name = "IVLIML"</code> or <code>"IVCF"</code> . <code>"C2SLS"</code> is a moment estimator and carries no correction parameter, so it is refused.
level	Size used for the reported verdict.

Details

The statistic. With $\hat{\rho}$ the vector of correlations between v and the reduced-form errors,

$$W = \hat{\rho}' [\widehat{\text{Var}}(\hat{\rho})]^{-1} \hat{\rho} \sim \chi^2(q),$$

q the number of endogenous regressors. With a single endogenous regressor it is exactly the squared z-ratio already printed for `rho_` in the fit.

Why this is an ordinary chi-square. ρ lives in the open unit ball, so $\rho = 0$ is an *interior* point of the parameter space. That is the opposite of the situation `inefficiency_test` faces, where $\sigma_u = 0$ sits on a boundary and the null distribution is a chi-bar-square mixture. No mixture is needed here.

Standard errors for ρ . `ivsfrm` parameterises the correlations as $\rho = t/\sqrt{1+t't}$, which keeps them inside the unit ball for any real t . Before version 1.2.0 the standard errors of ρ were reported as NA on the ground that this map has a non-diagonal Jacobian. It does, but the Jacobian is short –

$$\partial \rho_i / \partial t_j = \delta_{ij} / s - t_i t_j / s^3, \quad s = \sqrt{1+t't},$$

that is $J = (I - \rho\rho')/s$ – so $\text{Var}(\hat{\rho}) = J \text{Var}(\hat{t}) J'$ by the delta method. At the null $J = I$, which is what makes the test well calibrated there. `ivsfrm` now reports these standard errors and carries the full $\text{Var}(\hat{\rho})$ as `$vcov_rho`, which is what the *joint* test needs.

Reading a non-rejection. Failing to reject is not evidence for exogeneity. ρ is identified through the instruments, so weak instruments make $\hat{\rho}$ imprecise and the test powerless; the print method says so rather than letting the p-value speak alone.

Value

An object of class `c("sfa_endog_test", "htest")` carrying statistic (the Wald W), parameter (its degrees of freedom, the number of endogenous regressors), p. value, the estimated ρ with its standard errors and z-ratios, and `reject`.

References

- Hou, Z., Ramalho, J. J. S. and Roseta-Palma, C. (2025). Dealing with endogeneity in stochastic frontier models: A comparative assessment of estimators. *Energy Economics* 151, 108922.
- Amsler, C., Prokhorov, A. and Schmidt, P. (2016). Endogeneity in stochastic frontier models. *Journal of Econometrics* 190(2), 280-288.
- Kutlu, L. (2010). Battese-Coelli estimator with endogenous regressors. *Economics Letters* 109(2), 79-81.

See Also

`ivsfrm` for the fit, `inefficiency_test` for the boundary-null case, `spec_test` and `moment_range` for the distributional assumptions.

Examples

```
set.seed(9)
n <- 600
z <- rnorm(n); x1 <- rnorm(n); eps <- rnorm(n)
x2 <- z + eps
v <- 0.6 * eps + sqrt(1 - 0.36) * rnorm(n)
d <- data.frame(y = 0.5 * x1 + 0.5 * x2 + v - abs(rnorm(n)),
                x1 = x1, x2 = x2, z = z)

fit <- ivsfm(y ~ x1 + x2, endogenous = ~ x2, instruments = ~ z,
            data = d, model_name = "IVCF")
endogeneity_test(fit)
```

esfm

Stochastic frontier tolerating the wrong skewness

Description

Fits the extended stochastic frontier model of Hafner, Manner and Simar (2018), in which one parameter γ carries both the scale of inefficiency (its size) and the direction of skewness (its sign), so a wrongly skewed sample gives a well-defined non-degenerate fit instead of collapsing.

Usage

```
esfm(formula, data, dist = c("halfnormal", "exponential"), inefdec = TRUE,
     start = NULL)
```

```
symmetry_test(object)
```

Arguments

formula, data	Model and data, as for sfm .
dist	Base density for the inefficiency term.
inefdec	TRUE for a production frontier, FALSE for cost.
start	Optional numeric start, c(gamma, sigma_v, beta).
object	An "esfm" fit.

Details

The classical frontier implies a negatively skewed composed error. When the sample skewness comes out positive – common at small n or large σ_v/σ_u , and a small-sample accident rather than evidence against the model – the MLE collapses: σ_u is exactly zero, every firm is fully efficient, and the fit is uninformative. That is the Type I failure [skewness_test](#) reports.

This model keeps the frontier and widens the distribution of u . For $\gamma > 0$, u has the classical density with scale $|\gamma|$; for $\gamma < 0$ that density is mirrored about zero, shifted right by $B = a_0|\gamma|$ and truncated to $[0, B]$, giving *negative* skewness for u and so positive skewness for the composed error.

The truncation constant a_0 is fixed, not estimated: it is chosen so $E[u] = k_1|\gamma|$ whichever sign γ takes, which keeps the model a one-parameter scale family and avoids the identification difficulties of bounded-inefficiency models. $a_0 = 1.3892032925$ for the half-normal and 1.5936242600 for the exponential.

The classical model is nested at $\gamma > 0$, and $\gamma = 0$ is an *interior* point rather than a boundary. So the likelihood ratio test of $H_0 : \gamma = 0$ – no inefficiency, symmetric errors – is an ordinary $\chi^2(1)$, not the chi-bar-square mixture that `inefficiency_test` needs for the classical model.

What it does on a wrongly skewed sample. On a 50-observation draw with positive residual skewness, `sfm(model_name = "NHN")` returns $\sigma_u = 0.0003$ and mean efficiency 0.99976 – every firm on the frontier. `esfm()` returns $\hat{\gamma} = -0.57$ with slopes still near their true values.

Size of the LR test, 2000 replications against the paper’s 100,000, on its design ($\sigma_v = 1, \gamma = 0$):

n	50	100	200	500	1000
this package, 5%	0.178	0.082	0.054	0.061	0.058
paper, 5%	0.152	0.076	0.061	0.054	0.053

The test is badly oversized below $n = 100$ and usable from $n = 200$, which is the paper’s own conclusion.

Efficiency bias, and a caveat. The paper reports smaller bias in mean efficiency than the classical model even when the population skewness has the correct sign. That reproduces *when inefficiency is not small*: at $\gamma = 0.5, \sigma_v = 0.25$ the bias in $E[\exp(-U)]$ is -0.0001 at $n = 100$ against the classical model’s +0.021. But at $\gamma = 0.3$, where 22% of samples are wrongly skewed, this model overshoots the other way: bias -0.040 against the classical +0.036, and it is the larger of the two at $n = 50$ (-0.080 against +0.044). A small true γ lets the extended model fit sizeable inefficiency of either sign to noise. **It is the better choice when the wrong skewness is the problem, not a free improvement everywhere.**

Value

An object of class "esfm" (and "sfareg"), whose out matrix reports gamma, sigv and the frontier coefficients, with Efficiency holding $E[\exp(-U) | W]$ for each firm.

References

- Hafner, C. M., Manner, H. and Simar, L. (2018). The “wrong skewness” problem in stochastic frontier models: A new approach. *Econometric Reviews* **37**(4), 380–400.
- Almanidis, P., Qian, J. and Sickles, R. C. (2014). Stochastic frontier models with bounded inefficiency. In *Festschrift in Honor of Peter Schmidt*.

See Also

`skewness_test`, `inefficiency_test`, `sfm`

Examples

```
set.seed(3)
n <- 200
lx1 <- rnorm(n, 1.5, 0.3); lx2 <- rnorm(n, 1.8, 0.3)
```

```

y <- 0.9 + 0.6 * lx1 + 0.5 * lx2 + rnorm(n, 0, 0.25) - abs(rnorm(n, 0, 0.5))
d <- data.frame(y, lx1, lx2)

f <- esfm(y ~ lx1 + lx2, data = d)
f
symmetry_test(f)

```

FinnishElec

FinnishElec

Description

Cross-sectional data on Finnish electricity distribution firms, including annual averages of expenditure and output measures over a four-year regulatory period.

Usage

```
data("FinnishElec")
```

Format

A data frame with 89 observations on the following 6 variables.

`id` a character vector containing a unique identifier for each distribution firm
`x` a numeric vector containing total expenditure (TOTEX*) (1000 Euros)
`y1` a numeric vector containing weighted energy transmitted through the network (GWh of 0.4 kV equivalents)
`y2` a numeric vector containing total length of the network (km)
`y3` a numeric vector containing total number of customers connected to the network
`z` a numeric vector containing the proportion of underground cables in the total network length.

Details

*TOTEX includes capital expenditure (CAPEX), controllable operational expenditure (OPEX), and estimated external cost of interruptions.

Source

Kuosmanen, T. (2012). 'Stochastic semi-nonparametric frontier estimation of electricity distribution networks: Application of the StoNED method in the Finnish regulatory model.' *Energy Economics*, 34(6), pp. 2189-2199. doi:10.1016/j.eneco.2012.03.005

Examples

```

data(FinnishElec)
plot(FinnishElec)

```

gof_test

*Goodness-of-fit test for the assumed inefficiency distribution***Description**

Tests the distributional assumption on u , holding the normality of v fixed, following Wang, Amsler and Schmidt (2011).

Usage

```
gof_test(object, data = NULL, test = c("ks", "chisq"),
         null = c("bootstrap", "asymptotic"), B = 199, cells = 10,
         seed = NULL)
```

Arguments

object	An "sfareg" fit from sfm .
data	The data the model was fitted to. <code>sfm()</code> does not retain it, so it must be supplied or be findable from the fit's call.
test	"ks", "chisq", or both.
null	"bootstrap" for the parametric bootstrap of Wang, Amsler and Schmidt, or "asymptotic".
B	Bootstrap replications. Each one refits the model.
cells	Number of equiprobable cells for the chi-square test.
seed	Optional seed for the bootstrap.

Details

The distribution of u is the assumption in this model least often defended and least often tested, and it is testable: with the noise distribution maintained, the assumed u implies a distribution for the composed error, so rejecting that is rejecting the assumed u .

The test is on ε , not on $\hat{u}$. Wang, Amsler and Schmidt point out that $\hat{u} = E[u \mid \varepsilon]$ is a monotonic function of ε , so the KS test is identical either way and the chi-square test is identical when the cells are defined conformably – but ε is much easier to work with. It is a mistake, not a diagnostic, to compare the observed spread of $\hat{u}$ with the assumed density of u : those are different distributions.

Parameter estimation cannot be ignored. Both statistics are evaluated at $\hat{\theta}$, which changes their null distributions. The default "bootstrap" handles this by copying the estimation step exactly: each replication draws a composed error from the fitted model, rebuilds the response, *refits*, forms its own residuals, and recomputes the statistic. "asymptotic" returns a $\chi^2(k - 1 - m)$ p-value for the chi-square statistic, which the authors note is **conservative** when evaluated at the MLE – that reference distribution belongs to the minimum-chi-square estimator. There is no asymptotic KS p-value here: with θ estimated the Kolmogorov distribution does not apply, and Bai's (2003) martingale transformation is not implemented, so NA is returned rather than a number that would look valid.

Size and power. Measured in this package over 200 replications with $B = 49$, half-normal data fitted as half-normal for size and exponential data fitted as half-normal for power:

	n	KS .10	KS .05	chi2 .10	chi2 .05
size	200	0.065	0.025	0.085	0.065
size	800	0.070	0.040	0.065	0.010
power	200	0.505	0.395	0.235	0.145
power	800	0.985	0.965	0.810	0.655
power	2000	1.000	1.000	0.990	0.985

Both tests hold their size, mildly conservatively. **The KS test dominates the chi-square test at every sample size**, which is the paper's own conclusion and the reason it is the better default: the chi-square statistic discards the within-cell information that KS uses. Power against the half-normal/exponential pair – one of the harder distinctions, since both are one-parameter and similarly shaped – is about 0.4 at $n = 200$ and essentially 1 by $n = 800$, so a failure to reject in a small sample is weak evidence.

Value

A data frame with one row per test: test, statistic, null, p.value and B (replications actually used).

References

- Wang, W. S., Amsler, C. and Schmidt, P. (2011). Goodness of fit tests in stochastic frontier models. *Journal of Productivity Analysis* **35**, 95–118.
- Bai, J. (2003). Testing parametric conditional distributions of dynamic models. *Review of Economics and Statistics* **85**, 531–549.

See Also

[spec_test](#), [pcomposed_model](#), [sfm](#)

Examples

```
d <- as.data.frame(data_gen_cs(N = 200, rand = 5, sig_u = 1, sig_v = 0.5,
                             cons = 0.5, beta1 = 0.5, beta2 = 0.5,
                             a = 5, mu = 0.1))
f <- sfm(y_pcs ~ x1 + x2, model_name = "NHN", data = d)

## Every bootstrap replication REFITS, so keep B small in an example; use
## B = 999 for a real test.
gof_test(f, data = d, B = 39, seed = 1)

gof_test(f, data = d, test = "chisq", null = "asymptotic")
```

hscore	<i>Hyvarinen score for a Normal–Half-Normal frontier fit</i>
--------	--

Description

Evaluates the Hyvarinen score used to choose the robustness tuning parameter of the maximum L_q -likelihood and minimum density-power divergence estimators, following Sugawara and Yonekura (2021). Lower is better.

Usage

```
hscore(e, sigma_v, sigma_u, c, stable = TRUE)
```

Arguments

e	Numeric vector of composed residuals $y - x'\beta$.
sigma_v, sigma_u	Positive scalars, the noise and inefficiency scale parameters.
c	Robustness tuning parameter, $c = 1 - q$ for MLqE and $c = \alpha$ for MDPD. $c = 0$ is the maximum likelihood endpoint.
stable	Logical. TRUE (default) evaluates in log space; FALSE uses the natural-scale expression.

Details

The score is

$$H(c) = n^{-1} \sum_i \{2D''(e_i) + D'(e_i)^2\}, \quad D = (f^c - 1)/c,$$

with derivatives taken with respect to the composed residual.

The score is evaluated in logarithms. The natural-scale expression contains the fitted density raised to the power $c - 2$; since $c < 1$ that exponent is close to -2 , so a single observation whose density underflows makes the score non-finite. The failure is silent and directional: the tuning candidates that remain evaluable are those closest to maximum likelihood, so a naive implementation reports too little robustness. Set `stable = FALSE` to reproduce that behaviour for comparison.

Value

A single numeric value, the mean observationwise score.

References

- Hyvarinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research* 6, 695–709.
- Sugawara, S. and Yonekura, S. (2021). On selection criteria for the tuning parameter in robust divergence. *Entropy* 23(9), 1147.
- Bernstein, D.H., Parmeter, C.F. and Wright, I.A. (2026). On Robust Estimation of the Stochastic Frontier Model. Working paper.

See Also

[hscore_select](#), [calibrate_c](#), [density_weights](#)

Examples

```
set.seed(1)
e <- rnorm(500, 0, 0.3) - abs(rnorm(500, 0, 0.6))
hscore(e, sigma_v = 0.3, sigma_u = 0.6, c = 0.20)

## the two evaluations agree wherever both are computable
hscore(e, 0.3, 0.6, 0.20) - hscore(e, 0.3, 0.6, 0.20, stable = FALSE)

## but only one of them survives an observation in the far tail
e2 <- c(e, -60)
hscore(e2, 0.3, 0.6, 0.15)           # finite
hscore(e2, 0.3, 0.6, 0.15, stable = FALSE) # not finite
```

hscore_select	<i>Select the robustness tuning parameter by Hyvarinen score</i>
---------------	--

Description

Refits a Normal–Half-Normal stochastic frontier at each candidate value of the robustness tuning parameter and returns the value minimising the Hyvarinen score of [hscore](#). The search is coarse-to-fine and includes the maximum-likelihood endpoint $c = 0$, so the criterion is free to decline robustification altogether.

Usage

```
hscore_select(object, method = c("mlqe", "psi", "mdpd"),
              range = c(0, 0.60), coarse = 0.01, fine = 0.001,
              half_width = 0.02, multistart = TRUE, verbose = FALSE)
```

Arguments

object	A fitted model from sfm with <code>model_name = "NHN"</code> , used for the data, the formula and the reference starting values.
method	Robust criterion. "psi" and "mdpd" are the same estimator at a common tuning value.
range	Numeric length-2 search interval.
coarse, fine	Grid steps for the two passes.
half_width	Half-width of the refinement window around the coarse minimum.
multistart	Logical; see Details.
verbose	Logical, print progress.

Details

Two failure modes are guarded here, both of which return a converged fit and a finite criterion value if they are not.

Silent grid loss. The criterion is evaluated in log space (see [hscore](#)). Evaluated naively it is non-finite wherever any fitted density underflows, which discards candidates non-randomly – those nearest maximum likelihood survive – and so understates the robustness the data call for.

Warm-start capture. Walking the grid upward and warm-starting each fit from the previous one is efficient but unguarded: if one fit falls into the degenerate $\sigma_u \rightarrow 0$ basin, every later fit inherits it. With `multistart = TRUE` each candidate is fitted from both the warm start and a fixed reference start, keeping the better objective.

Value

An object of class "sfa_hscore": the selected tuning parameter `c`, its criterion value `hscore`, the whole search path as a data frame, the refitted model `fit`, and `n_finite`, how many candidates were scorable out of how many tried.

Reading the result

A flat region around the minimum is not non-identification in the usual sense; it means the criterion discriminates weakly among nearby tuning values. The drop from $c = 0$ to the minimum is a descriptive score difference, not a test of maximum likelihood against a robust alternative. Where the selected value matters, compare it with the fixed calibration from [calibrate_c](#) and check its stability by resampling: the selection can be bimodal across resamples even when the full-sample criterion clearly prefers a robust solution.

References

- Sugasawa, S. and Yonekura, S. (2021). On selection criteria for the tuning parameter in robust divergence. *Entropy* 23(9), 1147.
- Bernstein, D.H., Parmeter, C.F. and Wright, I.A. (2026). On Robust Estimation of the Stochastic Frontier Model. Working paper.

See Also

[hscore](#), [calibrate_c](#), [density_weights](#), [sfm](#)

Examples

```
set.seed(3)
n <- 200
x <- runif(n, 1, 10)
y <- 1 + 0.5 * log(x) + rnorm(n, 0, 0.3) - abs(rnorm(n, 0, 0.6))
dat <- data.frame(y = y, x = log(x))
fit <- sfm(y ~ x, data = dat, model_name = "NHN")

sel <- hscore_select(fit, method = "mlqe")
sel
plot(sel$path$c, sel$path$hscore, type = "l",
```

```
xlab = "c", ylab = "Hyvarinen score")
abline(v = sel$c, lty = 3)
```

Indian

Indian

Description

Panel data on 34 paddy farmers from Aurepalle, India, collected over ten years (1975-76 to 1984-85). The panel is unbalanced: farmers are observed between 2 and 10 times. Includes farmer characteristics (age, schooling) and production variables (output, land, labor, bullocks, input costs).

Usage

```
data("Indian")
```

Format

A data frame with 273 observations (an unbalanced panel of 34 farmers over 10 years) on the following 10 variables.

`id` a numeric vector containing a unique identifier for each farmer

`yr` a numeric vector containing the year of the observation

`age` a numeric vector containing the age of the primary decision maker

`school` a numeric vector containing the number of years of schooling of the primary decision maker

`yvar` a numeric vector containing the natural logarithm of the total value of output (rupees)

`Lland` a numeric vector containing the natural logarithm of the total area of land operated (ha)

`PIland` a numeric vector containing the proportion of land that is irrigated

`Llabor` a numeric vector containing the natural logarithm of the total number of hours of hired and family labour used

`Lbull` a numeric vector containing the natural logarithm of the number of hours of bullock labour used

`Lcost` a numeric vector containing the natural logarithm of the value of inputs including fertilizer, manure, pesticides, machinery, etc.

Source

Battese, G.E. and Coelli, T.J. (1995) 'A model for technical inefficiency effects in a stochastic frontier production function for panel data', *Empirical Economics*, 20(2), pp. 325-332. doi:10.1007/BF01205442.

References

Battese, G.E. and Coelli, T.J. (1992) 'Frontier production functions, technical efficiency and panel data: With application to paddy farmers in India', *Journal of Productivity Analysis*, 3(1-2), pp. 153-169. doi:10.1007/BF00158774.

Examples

```
data(Indian)
```

inefficiency_test	<i>Test whether there is any technical inefficiency</i>
-------------------	---

Description

Tests H_0 : no technical inefficiency, i.e. $\gamma = 0$ in the Battese-Corra parameterization $\gamma = \sigma_u^2/(\sigma_u^2 + \sigma_v^2)$, following Section 3 of Coelli (1995).

Usage

```
inefficiency_test(object, test = c("lr_1sided", "lr", "wald", "m3t"),
  level = 0.05)
```

Arguments

object	An "sfareg" fit from sfm , with model_name = "NHN" or "NTN".
test	Which statistics to compute; any subset, all by default.
level	Significance level used for the reject column.

Details

The two tests most often reported are the two with the wrong size. H_0 puts γ on the boundary of the parameter space, so the usual asymptotics fail. The likelihood ratio statistic is not $\chi^2(1)$ but the Gouriéroux, Holly and Monfort (1982) mixture $\frac{1}{2}\chi^2(0) + \frac{1}{2}\chi^2(1)$, whose 5% critical value is 2.71 rather than 3.84; using the $\chi^2(1)$ value makes the naive LR test conservative. The Wald ratio $\hat{\gamma}/s_{\hat{\gamma}}$ is worse: it is not asymptotically standard normal at the boundary at all.

Measured in this package on 400 replications per cell, half-normal data with $\sigma_u = 0$ and a nominal 5% level:

n	LR one-sided	LR naive	Wald	M3T
100	0.060	0.030	0.230	0.055
200	0.060	0.033	0.203	0.050
400	0.055	0.033	0.217	0.045
800	0.060	0.035	0.182	0.062

The one-sided LR test and the third-moment test hold their size; the naive LR test rejects about half as often as it should; the Wald test rejects three to four times too often and does not improve with n . Under the alternative ($\sigma_u = \sigma_v = 1$) the one-sided LR test has the better power of the two valid tests at every sample size, which is Coelli's recommendation. **The Wald column's apparently high power is not comparable, because its size is wrong.**

For model_name = "NTN" the null restricts both γ and μ , and the mixture becomes $\frac{1}{4}\chi^2(0) + \frac{1}{2}\chi^2(1) + \frac{1}{4}\chi^2(2)$.

The "m3t" statistic is the same one [skewness_test](#) reports as test = "coelli"; it is repeated here so the four tests can be compared in a single table.

Value

A data frame with one row per test: test, statistic, null (the reference distribution), p.value and reject. The log-likelihoods under both hypotheses are attached as attributes.

References

Coelli, T. (1995). Estimators and hypothesis tests for a stochastic frontier function: A Monte Carlo analysis. *Journal of Productivity Analysis* **6**, 247–268.

Gourieroux, C., Holly, A. and Monfort, A. (1982). Likelihood ratio test, Wald test, and Kuhn-Tucker test in linear models with inequality constraints on the regression parameters. *Econometrica* **50**, 63–80.

See Also

[skewness_test](#), [spec_test](#), [sfm](#)

Examples

```
d <- data_gen_cs(N = 300, rand = 7, sig_u = 1, sig_v = 0.5, cons = 0.5,
                beta1 = 0.5, beta2 = 0.5, a = 5, mu = 0.1)
f <- sfm(y_pcs ~ x1 + x2, model_name = "NHN", data = as.data.frame(d))
inefficiency_test(f)
```

influence_sfa	<i>Which observations are driving this frontier fit?</i>
---------------	--

Description

Empirical influence function and case-influence measures for a fitted stochastic frontier. Answers *which observations move the estimates*, and *whether the specification lets any single observation move them without bound*.

Usage

```
influence_sfa(object, scale = TRUE)
```

Arguments

- | | |
|--------|---|
| object | An "sfareg" fit. It must retain its likelihood, so fit with keep_objective = TRUE; the score matrix cannot otherwise be built. |
| scale | Multiply the influence function by <i>n</i> , so that influence[i,] approximates the effect on the estimate of deleting observation <i>i</i> and is therefore on the same scale as the coefficients. Default TRUE. |

Details

The standard outlier rules do not transfer to this model. Least-median-of-squares and its relatives ignore the **asymmetry** of the composed error, so a genuinely large u_i – an inefficient firm, the thing being measured – reads as an outlier and is discarded, which defeats the analysis.

The influence function is the right object instead. Maximum likelihood is an M-estimator, so its influence function is a linear transformation of the score, $IF_i \propto I(\theta)^{-1}\psi(y_i, \theta)$, and the estimator is *B-robust* exactly when that is bounded. Both pieces are already available: `estfun` supplies the per-observation scores and `vcov` the inverse information.

Which sensitivity to read. The raw sup-norm depends on how the parameters happen to be scaled, so it is *not* comparable between models with different parameter vectors. On a clean sample of 200 it reads 72.6 for "NHN" and 1322.5 for "tHN", which appears to say the robust specification is far worse; all it reflects is tHN's extra ν on its own scale. `sensitivity_std` measures the same influence function in the information metric and is invariant to reparameterisation. On that scale, contaminating a single response takes "NHN" from 13.5 to 32.3 and "tHN" from 13.1 to 17.2 – Stead, Wheat and Greene's finding, and invisible in the raw number.

Neither number has an absolute threshold. The comparison to make is between specifications on the same data, not either number against a cut-off.

What to do about a large value. Nothing, by deletion. A large sensitivity is a property of the specification: under an unbounded influence function the next most extreme point simply replaces the one removed. The remedies are a specification with bounded influence – Stead, Wheat and Greene's Student's t noise term, `model_name = "tHN"` – or a robust divergence criterion, `sfm(robust = )`, whose tuning is chosen by `hscore_select` or `calibrate_c`.

What this cannot see. Like `density_weights`, it is computed at the fitted surface. An observation mis-recorded in a *regressor* can bend that surface toward itself and so appear unremarkable here. Pair this with a leverage check before concluding that an observation is not driving a result.

Value

An object of class "sfa_influence":

<code>influence</code>	$n \times p$ matrix, the empirical influence function.
<code>d</code>	Length- n case influence $n s_i' V s_i$, the same quadratic form as Cook's distance and invariant to reparameterisation.
<code>sensitivity_std</code>	$\sup_i \sqrt{d_i}$, the self-standardised sensitivity – the one to compare across models .
<code>sensitivity</code>	$\sup_i \ IF_i\ $, the raw sup-norm.
<code>max_abs_by_parameter</code>	Largest $ IF $ in each coordinate.

References

- Stead, A.D., Wheat, P. and Greene, W.H. (2023). Robust maximum likelihood estimation of stochastic frontier models. *European Journal of Operational Research*, 309(1), 188–201.
- Bernstein, D.H., Parmeter, C.F. and Wright, I.A. (2026). On Robust Estimation of the Stochastic Frontier Model. Working paper.

See Also

[density_weights](#), [hscore_select](#), [sfm](#), [sfareg-methods](#)

Examples

```
set.seed(3)
n <- 120
x <- runif(n, 1, 10)
d <- data.frame(y = 1 + 0.5 * log(x) + rnorm(n, 0, 0.3) - abs(rnorm(n, 0, 0.6)),
                x = log(x))

fit <- sfm(y ~ x, data = d, model_name = "NHN", keep_objective = TRUE)
influence_sfa(fit)

## Contaminate one response and compare specifications on the
## self-standardised scale.
d2 <- d; d2$y[1] <- d2$y[1] - 5
a <- sfm(y ~ x, data = d2, model_name = "NHN", keep_objective = TRUE)
b <- sfm(y ~ x, data = d2, model_name = "tHN", keep_objective = TRUE)
c(NHN = influence_sfa(a)$sensitivity_std,
  tHN = influence_sfa(b)$sensitivity_std)
```

ivsfm

*Stochastic Frontier Models with Endogenous Regressors***Description**

Fits the stochastic frontier model of Amsler, Prokhorov and Schmidt (2016) in which one or more regressors is endogenous, in the simultaneous-equation sense: correlated with the statistical noise v . Three estimators are provided – full information maximum likelihood, a two-step control function, and corrected 2SLS. With `uhet` the model becomes that of Amsler, Prokhorov and Schmidt (2017), in which environmental variables entering the inefficiency scale may themselves be endogenous.

Usage

```
ivsfm(formula, endogenous, instruments, data,
      model_name = c("IVLIML", "IVCF", "C2SLS"), uhet = NULL,
      inefdec = TRUE, maxit.bobyqa = 10000, maxit.psoptim = 1000,
      maxit.optim = 1000, start_val = FALSE, PSopt = FALSE,
      optHessian = TRUE, Method = "L-BFGS-B", verbose = FALSE,
      rand.psoptim = NULL)
```

Arguments

<code>formula</code>	The structural frontier, $y \sim x_1 + x_2$, listing all regressors including the endogenous ones.
----------------------	---

endogenous	One-sided formula naming the endogenous variables, $\sim x2$. Each must appear in formula or in uheter.
instruments	One-sided formula of the <i>excluded</i> instruments, $\sim w1 + w2$. The included exogenous regressors are added automatically, so only the outside instruments are listed here. There must be at least as many as there are endogenous variables.
data	A data.frame holding every variable in the four formulas.
model_name	"IVLIML" (the default), "IVCF" or "C2SLS"; see 'Details'.
uhet	Optional one-sided formula of environmental variables, giving $\sigma_{U,i} = \sigma_U \exp(q'_i \delta)$. Supplying it selects the 2017 model; omitting it gives the 2016 one. Variables may appear in both uheter and endogenous.
inefdec	TRUE (default) for a production frontier, where inefficiency is subtracted; FALSE for a cost frontier.
maxit.bobyqa, maxit.psoptim, maxit.optim	Iteration caps for the three optimizer stages. Ignored by "C2SLS", which is closed-form.
start_val	Name the starting-value vector in the returned object.
PSopt	Run the particle-swarm stage between BOBYQA and optim.
optHessian	Compute the Hessian, and with it the standard errors.
Method	Method passed to optim for the final stage.
verbose	Report optimizer progress.
rand.psoptim	Optional seed for the particle-swarm stage.

Details

Unlike the rest of the package neither formula nor endogenous nor instruments takes a | segment. A pipe means "variance determinant" in `sfm` and `psfm`, and reusing it here for **ivreg**-style instrument syntax would give the same character two meanings across entry points, so a pipe is an error rather than being reinterpreted.

The model. Writing p_i for the endogenous block and z_i for the instruments,

$$y_i = \beta' x_i + v_i - u_i, \quad p_i = \Pi' z_i + \xi_i$$

$$u_i = \sigma_{U,i} |U_i|, \quad U_i \sim N(0, 1), \quad (v_i, \xi'_i) \sim N(0, \Omega)$$

with u_i independent of (v_i, ξ_i) . Endogeneity is the correlation between ξ and v : the regressors are correlated with the *noise*, not with the inefficiency.

Conditioning v on ξ turns this into an ordinary normal-half normal frontier in a shifted residual with a smaller noise variance,

$$\mu_{c,i} = \Sigma_{v\xi} \Sigma_{\xi\xi}^{-1} \xi_i, \quad \sigma_c^2 = \sigma_V^2 - \Sigma_{v\xi} \Sigma_{\xi\xi}^{-1} \Sigma_{\xi v}$$

which is what makes the likelihood closed-form and needs no simulation.

The estimators.

"IVLIML" APS (2016) Equation (13): maximum likelihood over the frontier and the reduced form jointly, so Π and $\Sigma_{\xi\xi}$ are estimated alongside β . Efficient, and its standard errors are correct as reported.

"IVCF" The two-step control function of Kutlu (2010), described in APS (2016) Section 4.4. Π and $\Sigma_{\xi\xi}$ are fixed at their reduced-form least-squares values and only the frontier block is maximized. Cheaper, still consistent, but it discards the information about the reduced form carried in the frontier likelihood; its conventional standard errors understate the true ones unless $\Sigma_{v\xi} = 0$. Measured over 30 replications at $n = 1000$: with $\rho = 0.6$ the mean reported standard error on the endogenous coefficient is 0.912 of the Monte Carlo standard deviation, while "IVLIML"'s is 1.005; with $\rho = 0$ the two are identical, as the theory says they should be. Bootstrap, or use "IVLIML", when the standard errors matter.

"C2SLS" APS (2016) Section 4.1: 2SLS, then σ_U and σ_V from the second and third moments of the residuals, then the intercept corrected by $\sqrt{2/\pi} \hat{\sigma}_U$. The residuals use the *actual* endogenous regressors, not their fitted values, which the paper flags explicitly. Moment-based rather than likelihood-based, so the fit carries no \$opt and `logLik` returns NA with a warning – the same footing as psfm's "GTRE_SEQ1". It estimates no ρ , since it never models the correlation it corrects for.

Inefficiency. jlms implements APS (2016) Equations (14)–(15), which condition on ξ as well as on ε . Since ξ is correlated with v it carries information about u even though u is independent of ξ , so this is a strictly better predictor than the plain Jondrow et al. (1982) formula.

What is assumed, and what is not covered. u is assumed independent of (v, ξ) . The case where the reduced-form error is correlated with the inefficiency as well requires the copula likelihood of APS (2016) Section 4.5 and is *not* implemented; the paper notes there that two intuitive approaches to that case do not work.

Value

An object of class "sfareg". out is a $p \times 3$ matrix, one row per parameter, columns par, st_err and t-val. Parameters are the frontier coefficients, sigma_u, sigma_v, any delta_ terms from uheta, and one rho_ per endogenous variable (absent for "C2SLS"). Alongside the usual components:

Pi	Reduced-form coefficient matrix.
Sigma_xi	Reduced-form error covariance.
rho	Correlation of each reduced-form error with v .
sigma_c	The conditional noise standard deviation σ_c , which is smaller than sigma_v whenever $\rho \neq 0$.
b_2sls	The 2SLS coefficients, which seed the likelihood and are the estimate "C2SLS" corrects.
wrong_skew	Whether the 2SLS residuals are skewed the wrong way.
jlms, efficiency	$E[u_i \varepsilon_i, \xi_i]$ and $\exp(-jlms)$.
vcov_rho	The full $q \times q$ covariance of $\hat{\rho}$, which <code>endogeneity_test</code> needs for the joint Wald statistic.

Standard errors for ρ arrived in 1.2.0. They were previously reported as NA, on the ground that $\rho = t/\sqrt{1+t't}$ has a non-diagonal Jacobian and a delta-method value computed as though it were diagonal would be wrong. That much was true, but the Jacobian itself is short – $J = (I - \rho\rho')/s$ with $s = \sqrt{1+t't}$ – so $\text{Var}(\hat{\rho}) = J \text{Var}(\hat{t}) J'$ is available exactly. Checked against a numerical Jacobian in the tests, and validated end-to-end by the size of `endogeneity_test`, which is 0.046 at a nominal 5% over 1000 replications.

References

- Amsler, C., Prokhorov, A. and Schmidt, P. (2016). Endogeneity in stochastic frontier models. *Journal of Econometrics*, **190**(2), 280–288.
- Amsler, C., Prokhorov, A. and Schmidt, P. (2017). Endogenous environmental variables in stochastic frontier models. *Journal of Econometrics*, **199**(2), 131–140.
- Kutlu, L. (2010). Battese-Coelli estimator with endogenous regressors. *Economics Letters*, **109**(2), 79–81.

See Also

[endogeneity_test](#) for the Wald test of whether the correction was needed at all; [sfm](#) for the frontier without one.

Examples

```
set.seed(1)
n <- 600
w1 <- rnorm(n); w2 <- rnorm(n); x1 <- rnorm(n)
eta <- rnorm(n)
v <- 0.5 * (0.6 * eta + sqrt(1 - 0.6^2) * rnorm(n))
x2 <- 0.9 * w1 - 0.7 * w2 + 0.5 * x1 + eta
y <- 0.5 + 0.8 * x1 - 0.6 * x2 + v - abs(rnorm(n))
dat <- data.frame(y = y, x1 = x1, x2 = x2, w1 = w1, w2 = w2)

fit <- ivsfm(y ~ x1 + x2, endogenous = ~x2, instruments = ~ w1 + w2,
            data = dat)

fit$out

## Ignoring the endogeneity biases the coefficient on x2 toward zero.
sfm(y ~ x1 + x2, model_name = "NHN", data = dat)$out
```

lcsfm

Latent Class Stochastic Frontier Models

Description

Fits the latent class stochastic frontier model of Greene (2005) and Orea and Kumbhakar (2004), in which several unobserved technologies coexist in one sample and each firm contributes to every one of them, weighted by its class probability.

Usage

```
lcsfm(formula, model_name = c("LCM", "LCM_Z", "LCM_CN"),
      data, n_class = 2, maxit.bobyqa = 10000, maxit.psoptim = 1000,
      maxit.optim = 1000, REPORT = 1, trace = 0, pgtol = 0,
      start_val = FALSE, PSopt = FALSE, optHessian = TRUE,
      inefdec = TRUE, penalty_c = 0, upper = NA, Method = "L-BFGS-B",
      verbose = FALSE, rand.psoptim = NULL)
```

Arguments

formula	a symbolic description for the model to be estimated. For "LCM_Z" the second part gives the covariates driving class membership ($y \sim x \mid z$).
model_name	Which specification to fit. "LCM" holds the class probabilities constant across the sample and lets every parameter vary by class; "LCM_Z" is "LCM" with the class probabilities depending on covariates supplied in the second part of the formula. "LCM_CN" is the contaminated normal frontier, in which only the <i>noise scale</i> varies between components and the frontier, the inefficiency scale and the class probabilities are all common – see 'Details'. Matching ignores case, so "lcm", "LCM" and "Lcm" are the same choice.
data	A data frame containing the variables named in formula.
n_class	Integer. The number of latent classes J , at least 2. Defaults to 2. The parameter count grows as $J(k+2)$ plus $(J-1)q$, so large J needs a correspondingly large sample; see 'Details'.
maxit.bobyqa	Maximum number of iterations for the bobyqa optimization routine
maxit.psoptim	Maximum number of iterations for the psoptim optimization routine
maxit.optim	Maximum number of iterations for the optim optimization routine
REPORT	reporting parameter
trace	Integer. Tracing level passed through to the optimizer; larger values print more. 0 (the default) is silent.
pgtol	Numeric. Projected-gradient tolerance passed to <code>optim()</code> 's "L-BFGS-B" method. 0 uses the optimizer's own default.
start_val	starting value (optional)
PSopt	use psoptim optimization routine (T or F)
optHessian	Logical. Should a numerically differentiated Hessian matrix be returned while using the optim routine? (for optim routine)
inefdec	Production or cost function
penalty_c	Tuning constant for Chen <i>et al.</i> 's (2001) penalty $c \log(J^J \prod_j p_j)$, which is added to the log-likelihood so that no class probability can be driven to 0 or 1. The default 0 leaves the ordinary likelihood untouched. A positive value is what lcsfm_homogeneity needs, and it changes what is maximised: <code>\$opt\$value</code> then holds the penalised objective, while <code>\$logLik_unpenalised</code> and <code>\$penalty</code> carry the two pieces separately. Only available for <code>model_name = "LCM"</code> , since the penalty is defined on a scalar mixing proportion.
upper	Vector of upper values for the optim package.
Method	The method to be used for optim. See 'Details' within optim.
verbose	Logical. Print optimization progress messages? Default is FALSE.
rand.psoptim	Integer. seed for replication of psoptim. Default to NULL.

Details

"LCM_CN", **the contaminated normal frontier**. A different use of the same machinery: instead of separating technologies, it gives the *noise* heavier tails. The noise density is a scale mixture of normals, $f_v(v) = \sum_j p_j \phi(v/\sigma_{vj})/\sigma_{vj}$, while the frontier, σ_u and the mixing proportions stay common across components. At $J = 2$ this is the scale-contaminated normal, an alternative to a Student- t noise term.

It has a closed form, which is why it is cheap. Since $f_\epsilon(e) = \int f_v(e + Su)f_u(u) du$ is linear in f_v , a mixture noise density passes straight through:

$$f_\epsilon(e) = \sum_j p_j f_{NHN}(e; \sigma_{vj}, \sigma_u)$$

a mixture of ordinary normal/half-normal densities sharing one σ_u . No extra integration is required, and the identity is verified against direct numerical integration in the tests.

The parameter vector is $(\sigma_{v1}, \dots, \sigma_{vJ}, \sigma_u, \beta, \text{logit})$ rather than one block per class.

It is also the specification `lcsfm_homogeneity`'s asymptotic null is stated for: one scalar parameter differing between components. On "LCM", where everything varies, the $\chi^2_{0:1}$ null rejected 63.5% of the time at a nominal 5% under a true null; on "LCM_CN" the same null gives 8.0%. Still mildly liberal, and the test says so, but usable.

The latent class stochastic frontier supposes that J distinct technologies are present in one sample and that which technology a firm operates is unobserved. Each class j has its own frontier and its own two scales,

$$y_i = x_i' \beta_j + v_{ij} - u_{ij}, \quad v_{ij} \sim N(0, \sigma_{vj}^2), \quad u_{ij} \sim N^+(0, \sigma_{uj}^2),$$

and class membership follows a multinomial logit,

$$P(j | q_i) = \exp(q_i' \delta_j) / \sum_m \exp(q_i' \delta_m), \quad \delta_J = 0,$$

with class J the reference. Every firm contributes to every class, weighted by its class probability, so the log-likelihood mixes the J composed densities rather than assigning firms to groups:

$$\log L_i = \log \sum_j P(j | q_i) f_j(\varepsilon_{ij}).$$

Under "LCM" the q_i are a constant, so the class probabilities are the same for every firm; under "LCM_Z" they are the covariates in the second part of the formula.

This is a different object from fitting separate frontiers to known subgroups: the groups are not known, and the uncertainty about them is carried through into the efficiency predictions. It is also more general than `zsfm`'s zero-inefficiency model, which is the restricted two-class case in which one class has no inefficiency at all.

`post.prob` is an $n \times J$ matrix of posterior class probabilities, `j1ms_class` holds the class-conditional Jondrow et al. (1982) predictions, and `j1ms` is their posterior-weighted average – a firm's inefficiency is only defined relative to a frontier, and which frontier it faces is itself uncertain, so reporting the modal class alone would discard that. `class` gives the modal class and `class_prob` the average prior class shares.

Label switching. Class labels in a finite mixture are identified only up to permutation: relabelling the classes and permuting δ gives the identical likelihood. Starting values are built by splitting the OLS residuals at their J -quantiles and refitting within each group, which makes a run reproducible on the same data – and avoids the saddle point that identical starting components would sit at, since equal posterior class probabilities give a zero score with respect to the class split – but it does not pin the labels to any external ordering. Do not compare class 1 across two fits, or against a simulation's true labels, without matching the classes first.

Choosing J . The likelihood cannot fall when a class is added, and the usual regularity conditions for a likelihood-ratio test of J against $J + 1$ do not hold, since the null puts a class probability on the boundary. Use `AIC()`/`BIC()`, the separation of the fitted frontiers, and whether the extra class is economically interpretable, rather than a p-value.

Value

An object of class "sfareg" containing the following components:

<code>out</code>	A matrix with parameter estimates, standard errors, and t-values. One row per parameter, named <code><par>_class<j></code> for the class blocks and <code>logit_<var>_class<j></code> for the class-probability coefficients.
<code>opt</code>	A list containing the optimization results from the final optimization procedure.
<code>total_time</code>	The total computation time for model estimation.
<code>start_v</code>	The starting values used in the optimization.
<code>model_name</code>	The name of the model estimated ("LCM" or "LCM_Z").
<code>formula</code>	The formula used in the model specification.
<code>jlms</code>	Posterior-weighted Jondrow et al. (1982) inefficiency predictions.
<code>post.prob</code>	An $n \times J$ matrix of posterior class probabilities, whose rows sum to one.
<code>jlms_class</code>	An $n \times J$ matrix of class-conditional JLMS predictions.
<code>class</code>	Integer vector giving each observation's modal class.
<code>class_prob</code>	Average prior class probabilities, one per class.
<code>n_class</code>	The number of classes fitted.
<code>coefficients</code>	A vector of estimated parameters.
<code>std.errors</code>	A vector of standard errors for the estimated parameters (NA if <code>optHessian = FALSE</code>).
<code>t.values</code>	A vector of t-values for the estimated parameters (NA if <code>optHessian = FALSE</code>).
<code>call</code>	The matched call.

Note

Standard errors require `optHessian` set to TRUE

Author(s)

Christopher F. Parmeter and David H. Bernstein

References

- Greene, W. (2005) 'Reconsidering heterogeneity in panel data estimators of the stochastic frontier model', *Journal of Econometrics*, 126(2), pp. 269-303. doi:10.1016/j.jeconom.2004.05.003.
- Orea, L. and Kumbhakar, S.C. (2004) 'Efficiency measurement using a latent class stochastic frontier model', *Empirical Economics*, 29(1), pp. 169-183. doi:10.1007/s00181-003-0184-2.
- Caudill, S.B. (2003) 'Estimating a mixture of stochastic frontier regression models via the em algorithm: a multiproduct cost function application', *Empirical Economics*, 28(3), pp. 581-598. doi:10.1007/s001810200142.
- Jondrow, J., Lovell, C.A.K., Materov, I.S. and Schmidt, P. (1982) 'On the estimation of technical inefficiency in the stochastic frontier production function model', *Journal of Econometrics*, 19(2-3), pp. 233-238. doi:10.1016/0304-4076(82)90004-5.

See Also

[lcsfm_homogeneity](#), [zsfm](#), [sfm](#), [psfm](#), [ttsfm](#), [sfareg-methods](#)

Examples

```
library(sfa)

## Two coexisting technologies, class probabilities constant.
set.seed(42)
n <- 400
x1 <- rnorm(n); x2 <- rnorm(n)
cls <- rbinom(n, 1, 0.4) + 1L
b <- rbind(c(1, 0.5, 0.5), c(3, 1.0, 0.2))
sv <- c(0.2, 0.2); su <- c(0.5, 1.0)
y <- b[cls, 1] + b[cls, 2] * x1 + b[cls, 3] * x2 +
  rnorm(n, 0, sv[cls]) - abs(rnorm(n, 0, su[cls]))

fit <- lcsfm(y ~ x1 + x2, model_name = "LCM",
            data = data.frame(y, x1, x2), n_class = 2)
fit$class_prob      # average class shares
head(fit$post.prob)  # posterior class membership, one row per firm
```

`lcsfm_homogeneity`

Test a latent class frontier against homogeneity

Description

Tests whether a `lcsfm` fit's latent classes are needed at all, against the null that one class describes the data. This is the first question a latent class fit raises and the ordinary likelihood ratio test cannot answer it, because under the null the second class's parameters are unidentified, the mixing proportion sits on a boundary, and the classes are interchangeable.

Usage

```
lcsfm_homogeneity(object, null = c("bootstrap", "chisq01"),
                  B = 199, c = 1, level = 0.05,
                  seed = NULL, quiet = FALSE, envir = parent.frame())
```

Arguments

object	An "sfareg" fit from <code>lcsfm</code> with <code>model_name = "LCM"</code> and <code>n_class >= 2</code> .
null	How to obtain the null distribution. "bootstrap" (the default) is a parametric bootstrap under the fitted one-class model. "chisq01" uses the published $\chi^2_{0:1}$ limit and warns : see 'Details' for why it does not apply to "LCM" as implemented.
B	Bootstrap replications. Each is two refits, so this dominates the running time. The smallest attainable p-value is $1/(B + 1)$.
c	Tuning constant in the penalty $c \log(J^J \prod_j p_j)$. Chen <i>et al.</i> (2001) report their simulations were insensitive to it; Stead, Wheat and Greene use 1 and 5.
level	Size used for the reported verdict.
seed	Optional integer seed for the bootstrap. The RNG state is saved and restored, so a seeded call does not disturb the caller's stream.
quiet	Set TRUE to suppress the bootstrap progress messages.
envir	Environment in which to re-evaluate the fit's data argument.

Details

The statistic. `lcsfm` is refitted maximising the *modified* likelihood – the ordinary log-likelihood plus Chen *et al.*'s penalty $c \log(J^J \prod_j p_j)$, which diverges to $-\infty$ as any class probability approaches 0 or 1 and so keeps every class occupied. The statistic is twice the difference between that maximum and the maximised log-likelihood of the one-class model, which is `sfm(model_name = "NHN")` on the same formula, since a single latent class *is* the ordinary frontier. The penalty is exactly zero at equal class probabilities, so the null model is recovered without a handicap.

Why the default null is a bootstrap. Stead, Wheat and Greene report the modified statistic is asymptotically $\chi^2_{0:1}$, a 50:50 mixture of a point mass at zero and χ^2_1 . That result is stated for the case in which a *scalar* parameter differs between classes – Zhu and Zhang (2004) require θ_1 and θ_2 scalar, and the paper's own application holds every parameter except the noise scale common across classes.

`lcsfm`'s "LCM" does not do that: it lets σ_v, σ_u and the whole frontier vector vary by class. The limit therefore does not carry over, and this was measured rather than assumed. Over 200 replications of a one-class data generating process at $n = 400$:

nominal size	10%	5%
actual rejection under "chisq01"	82.0%	63.5%

with the sampling distribution tracking χ^2_5 – exactly the parameter-count difference – rather than $\chi^2_{0:1}$, whose median is 0 against an observed 4.21.

A parametric bootstrap makes no appeal to an asymptotic distribution: it simulates from the fitted null and re-runs the whole procedure, so it is valid whatever the parameter structure. Over 60 replications of the same one-class process with $B = 99$:

nominal size	10%	5%
"chisq01"	82.0%	63.5%
"bootstrap"	10.0%	6.7%

both within a binomial standard error of nominal, with the p-values indistinguishable from uniform (KS against Uniform(0,1), $p = 0.80$). On a sample that "chisq01" rejected at $p = 0.005$, the bootstrap returns $p = 0.32$, and on genuinely two-class data the statistic is 17 times the largest of 99 null draws.

Cost. The bootstrap is $B + 1$ pairs of refits. Budget accordingly, and use seed for reproducibility.

"LCM_Z" is **refused**, by both this function and `lcsfm(penalty_c = )`: it makes the class probabilities depend on covariates, and the penalty is defined on a scalar mixing proportion.

Value

An object of class `c("sfa_mlrt", "htest")`, so it prints like any other R test. Alongside `statistic` and `p.value` it carries `null`, `B`, `boot` (the bootstrap null draws), `boot_ok`, `penalty_c`, `penalty_at_max`, the penalised and unpenalised J -class log-likelihoods, `logLik_1`, `class_prob`, and `reject`.

References

- Chen, H., Chen, J. and Kalbfleisch, J.D. (2001). A modified likelihood ratio test for homogeneity in finite mixture models. *Journal of the Royal Statistical Society B*, 63(1), 19–29.
- Stead, A.D., Wheat, P. and Greene, W.H. (2023). On hypothesis testing in latent class and finite mixture stochastic frontier models, with application to a contaminated normal-half normal model. *Journal of Productivity Analysis*, 60(1), 37–48.
- Zhu, H.-T. and Zhang, H. (2004). Hypothesis testing in mixture regression models. *Journal of the Royal Statistical Society B*, 66(1), 3–16.

See Also

[lcsfm](#), [TIC](#), [skewness_test](#)

Examples

```
set.seed(21)
n <- 300
x1 <- rnorm(n)
cls <- rbinom(n, 1, 0.5)
d <- data.frame(y = ifelse(cls == 1, 4, 1) + x1 +
                 rnorm(n, 0, 0.4) - abs(rnorm(n, 0, 0.6))),
               x1 = x1)

f <- lcsfm(y ~ x1, model_name = "LCM", data = d, n_class = 2)

## Small B for the example only; use the default in real work.
```

```
lcsfm_homogeneity(f, B = 19, seed = 1, quiet = TRUE)
```

marginal_effects

Marginal Effects of the Variance Determinants on Inefficiency

Description

For a model whose inefficiency scale depends on covariates (the `_Z` models), the marginal effects of those covariates on the mean and variance of the inefficiency term, $\partial E[u]/\partial z_k$ and $\partial \text{Var}[u]/\partial z_k$.

Usage

```
marginal_effects(object, average = FALSE, component = c("u", "h"))
```

Arguments

object	an object of class "sfareg" from <code>sfm</code> ("NHN_Z", "NE_Z") or <code>psfm</code> ("TRE_Z", "GTRE_Z"), fitted with a variance-determinant formula such as $y \sim x_1 + x_2 \mid z$.
component	which inefficiency component to differentiate. "u" (the default) is the only choice for every model except <code>psfm(model_name = "GTRE_Z")</code> , where it is the <i>transient</i> component; "h" selects that model's <i>persistent</i> component, parameterized by the third formula segment $y \sim x \mid z \mid z_p$. The output columns are named after the component, so a table cannot be misread once separated from the call that produced it.
average	if TRUE, return only the named vector of average marginal effects. Default FALSE, which returns the full per-observation table with that vector attached as the "average" attribute.

Details

A δ coefficient is not interpretable on its own: it sits in a log link for a scale parameter, so its units are neither those of u nor those of z , and its magnitude is not comparable across models whose link differs. The marginal effect is on the scale of u , and is what applied work reports.

Writing $s = \sigma_u(z)$, every derivative reduces to $\partial s/\partial z_k$ times a constant that cancels back into the moment itself. For the half-normal and the exponential alike:

$$\begin{aligned} \text{SD link: } \partial E[u]/\partial z_k &= \delta_k E[u], & \partial \text{Var}[u]/\partial z_k &= 2\delta_k \text{Var}[u] \\ \text{variance link: } \partial E[u]/\partial z_k &= (\delta_k/2) E[u], & \partial \text{Var}[u]/\partial z_k &= \delta_k \text{Var}[u] \end{aligned}$$

The factor of two matters. `sfm()`'s "NHN_Z" and "NE_Z" place the linear predictor on the standard deviation, $\sigma_u = \exp(z'\delta)$, while `psfm()`'s "TRE_Z" and "GTRE_Z" place it on the variance, $\sigma_u = \sqrt{\exp(z'\delta)}$. Reading a δ from one family with the other's convention in mind is wrong by exactly this factor. Reporting the marginal effect instead of the coefficient removes the trap, since the effect is on the scale of u either way.

The link is read from the fit rather than assumed, so a `sfm` model fitted with `z_link = "var"` and a `psfm` model are handled the same way and their effects are directly comparable.

`psfm(model_name = "GTRE_Z")` separates *persistent* inefficiency h_i from *transient* u_{it} and gives each its own determinant block, so both are available: `component = "u"` for the transient effects and `component = "h"` for the persistent ones. Each block is located by NAME rather than by position, because the σ_h coefficients are the trailing ones and a positional rule would report them under a σ_u label.

Effects are per observation, because σ_u varies with z . Constant columns of the z design are dropped: the derivative with respect to an intercept is not a marginal effect.

Standard errors are deliberately not reported. The effect is a nonlinear function of both δ and z , so its sampling distribution requires the delta method through the full covariance matrix or a bootstrap; a number printed without one would be read as inference.

Value

With `average = FALSE`, a data frame with one row per observation: `sigma_u`, `E_u`, `Var_u`, then one `dE_u.d<z>` and one `dVar_u.d<z>` column per non-constant variance determinant. Attributes "average" (the average marginal effects), "link" and "family" record the convention used.

With `average = TRUE`, just the named vector of average marginal effects.

See Also

`sfm`, `efficiency_ci`

Examples

```
dat <- data_gen_cs(N = 300, rand = 1, sig_u = 1, sig_v = 0.3,
                  cons = 0.5, beta1 = 0.5, beta2 = 0.5, a = 5, mu = 0.5)
fit <- sfm(y_pcs_z ~ x1 + x2 | z, data = dat, model_name = "NHN_Z")

me <- marginal_effects(fit)
head(me)

## The conventional summary.
marginal_effects(fit, average = TRUE)
```

meanefficiency

Model-Implied Mean Efficiency

Description

The mean efficiency implied by the fitted distribution of inefficiency, $E[\exp(-U)]$, and the mean efficiency among the most efficient p of that distribution.

Usage

```
meanefficiency(object,
  p = c(0.01, 0.05, 0.10, 0.25, 0.50, 0.75, 0.90, 0.95, 0.99),
  use_closed_form = TRUE)
```

Arguments

<code>object</code>	An "sfareg" fit.
<code>p</code>	Proportions in $(0, 1]$ at which to report supra-percentile mean efficiency. $p = 1$ is the whole distribution and returns the unconditional mean.
<code>use_closed_form</code>	Use the closed-form expression where one exists. Setting it to FALSE forces numerical integration of the same density, which is how the closed forms are checked.

Details

A different quantity from the reported average. `print` and `summary` report the average of the per-observation conditional predictions $E[\exp(-u_i) \mid \varepsilon_i]$. That is a property of the sample in hand. $E[\exp(-U)]$ is a property of the fitted *model*, and it is the number that can be compared across studies.

Closed forms. Available for the half-normal ($2e^{\sigma_U^2/2}\Phi(-\sigma_U)$), exponential ($1/(1 + \sigma_U)$), truncated normal, Rayleigh, gamma ($(1 + \theta)^{-k}$), uniform $((1 - e^{-\theta})/\theta)$ and generalized exponential ($2/((\sigma_U + 1)(\sigma_U + 2))$). The lognormal, Weibull and Nakagami have none in elementary terms and are integrated numerically; method in the returned list says which route was taken.

Every closed form is checked against numerical integration of the same density in the package's tests, and the density is separately checked against its own quantile function by simulation. The first catches an algebraic slip; the second catches a parameter being read off the fit incorrectly, which would otherwise make both routes agree and both be wrong.

Supra-percentile means are always integrated. Efficiency is decreasing in u , so the most efficient p are the smallest p of u : the reported value is $p^{-1} \int_0^{Q(p)} e^{-u} f(u) du$. It necessarily exceeds the unconditional mean and falls as p rises.

Value

A list with `model`, `distribution`, the parameters used, `mean_efficiency`, the method used to obtain it, and `supra`, a data frame of p against mean efficiency.

See Also

[efficiency](#) for the per-observation predictors.

Examples

```
d <- data_gen_cs(N = 400, rand = 1, sig_u = 1, sig_v = 0.3, cons = 0.5,
  beta1 = 0.5, beta2 = 0.5, a = 1, mu = 0.5)
fit <- sfm(y_pcs ~ x1 + x2, model_name = "NHN", data = d)
```

```
r <- meanefficiency(fit)
r$mean_efficiency
r$supra
```

moment_range	<i>What skewness and excess kurtosis can a pair of distributions produce?</i>
--------------	---

Description

Reports, for each pair of noise and inefficiency distributions, the set of skewness and excess kurtosis values the composed error $\varepsilon = v \pm u$ can attain at all, and says which pairs the residual moments rule out. This is the Type II failure check of Papadopoulos and Parmeter (2021): a pair whose attainable set does not contain the sample moments is refuted without any test statistic. It needs no frontier fit.

Usage

```
moment_range(object = NULL,
              noise = c("normal", "logistic", "laplace", "uniform", "t"),
              inefficiency = c("halfnormal", "exponential", "genexponential",
                               "truncatednormal", "gamma"),
              inefdec = NULL, df = 8)
```

Arguments

object	An "sfareg" fit, whose OLS residuals are used, or a numeric vector of residuals. Omit it to get the attainable ranges alone, as a reference table.
noise	Assumed distributions of v ; all five by default. Only the excess kurtosis of v enters.
inefficiency	Assumed distributions of u ; all five by default.
inefdec	Orientation. Taken from the fit when there is one, otherwise production. A cost frontier puts the skewness on the positive side.
df	Degrees of freedom for noise = "t", whose excess kurtosis is $6/(\nu - 4)$. Must exceed 4.

Details

The two equations. With $R = \text{SNR}^2/(1 + \text{SNR}^2)$ the share of composed-error variance contributed by u , Papadopoulos and Parmeter (2021, eqs. 5 and 10) give

$$|\gamma_1(\varepsilon)| = \gamma_1(u) R^{3/2}, \quad \gamma_2(\varepsilon) = \gamma_2(v)(1 - R)^2 + \gamma_2(u) R^2.$$

Because R lies in $(0, 1)$, the composed skewness can never exceed the skewness of u itself, whatever the noise: the Normal-Half-Normal pair is confined to $\gamma_1(\varepsilon) \in (-0.995, 0)$ and the Normal-Exponential pair to $(-2, 0)$. The kurtosis equation is quadratic in R , so its range is not simply

the interval between the two components' values – with leptokurtic noise it dips below *both* before rising, reaching $\gamma_2(v)\gamma_2(u)/(\gamma_2(v) + \gamma_2(u))$ at $R = \gamma_2(v)/(\gamma_2(v) + \gamma_2(u))$.

Which u can fail. Half-normal, exponential and generalized exponential have fixed skewness and kurtosis, so their attainable sets are the narrowest. The truncated normal is indexed by μ/σ_u and sweeps $\gamma_1(u) \in (0, 2)$ and $\gamma_2(u) \in (-0.243, 6)$ – it is the only one here whose composed error can be platykurtic under normal noise. The gamma has $\gamma_1(u) = 2/\sqrt{k}$ and $\gamma_2(u) = 6/k$, unbounded as $k \rightarrow 0$, so a gamma inefficiency term can *never* suffer a Type II failure; the price, as PP2021 note, is that $k < 1$ makes the density log-convex.

This is a diagnostic, not a test. PP2021 section 2.3 shows the range check over-rejects, and the package's own replication of their Tables 1-5 confirms it: with correctly specified Normal-Half-Normal data at $n = 200$ and $\text{SNR} = 3$, the sample excess kurtosis falls outside its theoretical range 26% of the time, and the sample skewness 16% of the time. Use [spec_test](#) for the formal statistic, which is built on the same two equations but studentised and bootstrapped.

Relation to the wrong-skew problem. A Type I failure – residual skewness of the wrong sign, Waldman (1982) – is a property of the data and is reported for every row at once. Type II failures are pair-specific: they are what tells you that the residuals are too skewed, or too heavy-tailed, for the particular u you assumed. Wrong skew is handled by [skewness_test](#) and, as an estimator, by [esfm](#).

Value

A data frame of class "sfa_moment_range", one row per pair, with the attainable interval for the composed skewness (g1_lo, g1_hi) and excess kurtosis (g2_lo, g2_hi), logical skew_ok and kurt_ok, and a verdict of "ok", "type I (wrong skew)" or "type II (...)". The sample moments, sample size and orientation are attributes.

References

- Papadopoulos, A. and Parmeter, C. F. (2021). Type II failure and specification testing in the Stochastic Frontier Model. *European Journal of Operational Research* 293(3), 990-1001.
- Waldman, D. M. (1982). A stationary point for the stochastic frontier likelihood. *Journal of Econometrics* 18(2), 275-279.

See Also

[spec_test](#) for the formal test built on the same two equations, [skewness_test](#) for the Type I failure, [esfm](#) for an estimator that tolerates it.

Examples

```
## The reference table on its own: no data needed.
moment_range(noise = "normal")

set.seed(4)
d <- data_gen_cs(N = 400, rand = 4, sig_u = 1, sig_v = 0.5, cons = 1,
                 beta1 = 0.5, beta2 = 0.5, a = 1, mu = 0.5)

## Exponential inefficiency, read through a normal-half-normal lens: the
## residuals are more skewed than a half-normal can make them.
```

```
fit <- sfm(y_pcs_e ~ x1 + x2, data = d, model_name = "NHN")
moment_range(fit, noise = "normal")
```

npsfm

Nonparametric Stochastic Frontier Models

Description

Fits a stochastic frontier whose frontier function is estimated by kernel regression rather than assumed linear. Two estimators are available: the two-step estimator of Fan, Li and Weersink (1996), and the local method-of-moments estimator of Simar, Van Keilegom and Zelenyuk (2017), which additionally lets both variance components vary with the covariates.

Usage

```
npsfm(formula, data, method = c("FLW", "SVKZ", "PSZ", "KPST", "MY", "SZ"),
      dist = c("hn", "exp", "gamma", "unif"),
      regtype = c("lc", "ll"), bw.sel = c("cv.ls", "cv.aic"),
      bw = NULL, cost = FALSE, eff = TRUE,
      maxit = 5000, tol = 1e-3, iter = 25,
      rts = c("vrs", "crs", "drs", "irs"),
      prior.fit = NULL, log.form = TRUE, verbose = FALSE)
```

Arguments

formula	A single-part symbolic description of the frontier, $y \sim x_1 + x_2$. Unlike the package's parametric entry points <code>npsfm()</code> takes no <code> z</code> segment: heteroskedasticity is handled nonparametrically through the covariates themselves under <code>method = "SVKZ"</code> , and a pipe is an error rather than something silently ignored.
data	A data frame containing the variables named in <code>formula</code> .
method	Which estimator to use. "FLW" Fan, Li and Weersink (1996). Estimates $E[y x]$ by kernel regression, then recovers the scale parameters from the residuals – by maximizing their concentrated likelihood in λ when <code>dist = "hn"</code>, and by inverting central moments otherwise. σ_u and σ_v are constants; only the frontier is smooth. "SVKZ" Simar, Van Keilegom and Zelenyuk (2017). Runs three local-linear regressions – y on x, then the squared and cubed residuals on x – and inverts the local moments, so $\sigma_u(x)$ and $\sigma_v(x)$ both vary with the covariates. No optimizer runs. Normal-half normal only. "PSZ" (or "KPST") Park, Simar and Zelenyuk. Local <i>maximum likelihood</i>: at every evaluation point the frontier and both log variance components are given local-linear expansions and the kernel-weighted normal-half normal likelihood is maximized in those $3(k + 1)$ parameters. One optimization per observation, so it is much slower than the two above. $\sigma_u(x)$ and $\sigma_v(x)$ vary with the covariates.

	"MY" Martins-Filho and Yao. Iterative local likelihood: alternates a local-linear fit of the frontier at every evaluation point, holding (λ, σ) fixed, with a global update of (λ, σ) from the resulting composed residuals, until the scale parameters stop moving. Cost is (observations $\times$ iterations) optimizations. σ_u and σ_v are constants; only the frontier is local. "SZ" Simar and Zelenyuk (2011). Not an estimator in its own right: it takes an already-estimated smooth frontier and passes its fitted values through an output-oriented DEA, imposing the monotonicity and (under <code>rts = "vrs"/"crs"</code>) convexity that a kernel fit does not guarantee. Supply the prior fit through <code>prior.fit</code>, or leave it NULL to fit "SVKZ" first. The DEA is solved internally as one linear program per unit and needs the lpSolve package; it is implemented for production frontiers only.
	Matching ignores case, so "nhn", "NHN" and "Nhn" are the same choice.
<code>dist</code>	Distribution of the one-sided inefficiency term, for <code>method = "FLW"</code> only: "hn" (half normal, the default), "exp" (exponential), "gamma", or "unif" (uniform on $[0, b]$). Every other method is derived for the half-normal case and errors for anything else.
<code>regtype</code>	Kernel regression type passed to <code>np::npregbw</code> : "lc" (local constant, the default) or "ll" (local linear). Applies to "FLW"; "SVKZ" is local linear throughout by construction.
<code>bw.sel</code>	Bandwidth selection method: "cv.ls" (least-squares cross-validation, the default) or "cv.aic". Ignored when <code>bw</code> is supplied.
<code>bw</code>	Optional numeric vector of bandwidths, one per covariate. When supplied, cross-validation is skipped and these are used directly – useful for undersmoothing, for sensitivity analysis, or simply to avoid paying for bandwidth selection repeatedly in a simulation.
<code>cost</code>	Logical. FALSE (the default) fits a production frontier, in which inefficiency is subtracted; TRUE fits a cost frontier.
<code>eff</code>	Logical. Compute observation-level inefficiency predictions (<code>u_hat</code> , <code>exp_u_hat</code>). Available for <code>dist = "hn"</code> and <code>dist = "exp"</code> ; the gamma and uniform cases return moment estimates only. Defaults to TRUE.
<code>maxit</code>	Maximum function evaluations for each local optimization under "PSZ" (passed to <code>minqa::bobyqa</code>) and maximum iterations for each local fit under "MY" (passed to <code>optim</code>). Ignored by the other methods. Defaults to 5000. Raise it if convergence reports many non-zero codes.
<code>tol</code>	Convergence tolerance for "MY": the iteration stops when the squared change in (λ, σ) falls below this. Defaults to $1e-3$.
<code>iter</code>	Maximum number of outer iterations for "MY". Defaults to 25.
<code>rts</code>	Returns-to-scale assumption for the DEA step under "SZ": "vrs" (the default), "crs", "drs" or "irs". These restrict $\sum_j \lambda_j$ to be unrestricted, $= 1$, ≤ 1 and ≥ 1 respectively.
<code>prior.fit</code>	For "SZ": a numeric vector of already-estimated frontier values, one per observation, to be monotonized. If NULL (the default) an "SVKZ" fit is computed first and used.

log.form	For "SZ": whether the data are in logs, as is conventional in this literature. When TRUE (the default) the frontier and covariates are exponentiated before the DEA step and the result is returned to the log scale.
verbose	Logical. Report progress through the per-observation loops of "PSZ" and the outer iterations of "MY". Defaults to FALSE.

Details

Both estimators relax the parametric frontier of `sfm` while keeping the composed-error structure $y = m(x) + v - u$. Because a kernel regression of y on x estimates $E[y|x] = m(x) - E[u]$ rather than $m(x)$, both proceed by fitting that conditional mean and then shifting it back up by an estimate of $E[u]$.

Where they differ. "FLW" treats σ_u and σ_v as constants, so its correction is a single number and the fitted gradients are those of the conditional mean. "SVKZ" estimates $\sigma_u(x)$ from the local third moment, so the correction varies across observations and the frontier gradient picks up an extra term through the chain rule.

Least squares versus local likelihood. "FLW" and "SVKZ" both begin from a least-squares kernel regression, which estimates $E[y|x] = m(x) - E[u]$ and therefore needs the mean shift described above. "PSZ" and "MY" instead maximize the *composed-error* likelihood locally, in which the local intercept is $m(x)$ directly and no shift is applied. They pay for that with one numerical optimization per observation – for "MY", per observation *per iteration* – so expect them to be one to two orders of magnitude slower than "FLW". Both are seeded from an "FLW" fit.

Which to use. On a simulated nonlinear frontier with $\sigma_u = 0.6$, $\sigma_v = 0.25$ (3 replications), mean absolute frontier error at $n = 300$ was 0.070 for "MY", 0.079 for "FLW", and 0.116 for both "SVKZ" and "PSZ". "FLW" is the steadiest for its cost; "MY" is the most accurate if the run time is acceptable; "SVKZ" and "PSZ" earn their keep when σ_u genuinely varies with x , which this design does not test.

Wrong skew. The identification of σ_u rests on the residuals being negatively skewed. "FLW" inverts a single sample moment and either succeeds or warns. "SVKZ" inverts a *local* third moment, which is far noisier, and at any point where the estimated skew has the wrong sign the implied $\sigma_u(x)^3$ is negative; following the paper those points are floored at $\sigma_u(x) = 0$ and their contribution to the frontier gradient is set to zero. `wrong.skew` records which observations these were. A large share of them means the local third moment is too noisy to be informative and the "SVKZ" fit should not be trusted – "FLW" is much steadier at moderate sample sizes.

The np dependency. Both estimators need kernel regression and bandwidth selection from the `np` package, which is listed under `Suggests` rather than `Imports` because nothing else in `sfa` requires it. `npsfm()` checks for it and stops with an install instruction if it is missing. Bandwidth selection by cross-validation is the dominant cost and scales quadratically in the sample size; supply `bw` to skip it.

Value

An object of class "npsfareg". This is deliberately *not* an "sfareg" object: there is no parameter vector with standard errors, so `coef()`, `vcov()` and `logLik()` would have nothing meaningful to return. `fitted()`, `residuals()`, `nobs()`, `print()` and `summary()` are provided. Components:

frontier	The estimated frontier $\hat{m}(x)$, i.e. the kernel fit shifted up by the estimated $E[u]$.
----------	--

frontier.grad	Matrix of estimated frontier gradients, one row per observation and one column per covariate.
conditional.mean	The uncorrected kernel fit of $E[y x]$, before the mean shift.
residuals	Composed residuals measured against the corrected frontier, $y - \hat{n}(x)$. These are negative up to noise, unlike the centered residuals of the underlying kernel regression.
mean.correction	The estimated $E[u]$: a scalar for "FLW", a vector for "SVKZ".
sigma.u, sigma.v	Estimated scale parameters. Scalars under "FLW"; vectors of $\sigma_u(x_i)$, $\sigma_v(x_i)$ under "SVKZ".
lambda, sigma	Returned by "FLW" with dist = "hn" only: $\lambda = \sigma_u/\sigma_v$ and $\sigma = \sqrt{\sigma_u^2 + \sigma_v^2}$.
theta	Returned by "FLW" with dist = "exp" or "gamma": the rate parameter of the one-sided term.
b	Returned by "FLW" with dist = "unif": the estimated upper bound of the uniform.
sigma.u.grad, wrong.skew	Returned by "SVKZ": the gradient of $\sigma_u(x)$, and a logical vector flagging observations whose local third moment had the wrong sign. "PSZ" also returns sigma.u.grad and sigma.v.grad, the local-linear slopes of the two log variance functions.
convergence	Returned by "PSZ": the minqa::bobyqa status code from each observation's local optimization, 0 for success. A large share of non-zero codes means maxit is too low and the local fits should not be trusted.
iterations, converged, tol.reached	Returned by "MY": how many outer iterations ran, whether the tolerance was met, and the final squared change in (λ, σ) .
prior.fit, dea.efficiency, rts	Returned by "SZ": the smooth frontier that was monotonized, the DEA efficiency scores, and the returns-to-scale assumption used.
u_hat, exp_u_hat	Jondrow et al. (1982) inefficiency predictions $E[u \varepsilon]$ and Battese-Coelli (1988) efficiency predictions $E[\exp(-u) \varepsilon]$. Returned when eff = TRUE and dist is "hn" or "exp".
bws	The bandwidth object(s) used: one for "FLW", a list of three (r1, r2, r3) for "SVKZ".
method, dist, formula, call, cost, regtype, bw.sel, nobs, total_time, data	Settings and bookkeeping.

Note

Requires the **np** package, which is in Suggests. Bandwidth cross-validation dominates the run time and scales quadratically in the sample size.

Author(s)

Christopher F. Parmeter and David H. Bernstein

References

Fan, Y., Li, Q. and Weersink, A. (1996) 'Semiparametric estimation of stochastic production frontier models', *Journal of Business & Economic Statistics*, 14(4), pp. 460-468.

Simar, L., Van Keilegom, I. and Zelenyuk, V. (2017) 'Nonparametric least squares methods for stochastic frontier models', *Journal of Productivity Analysis*, 47(3), pp. 189-204.

Jondrow, J., Lovell, C.A.K., Materov, I.S. and Schmidt, P. (1982) 'On the estimation of technical inefficiency in the stochastic frontier production function model', *Journal of Econometrics*, 19(2-3), pp. 233-238.

Battese, G.E. and Coelli, T.J. (1988) 'Prediction of firm-level technical efficiencies with a generalized frontier production function and panel data', *Journal of Econometrics*, 38(3), pp. 387-399.

See Also

[sfm](#) for parametric cross-sectional frontiers, [psfm](#) for panel models, and [data_gen_cs](#) for simulating data with known true parameters.

Examples

```
if (requireNamespace("np", quietly = TRUE)) {
  set.seed(42)
  n <- 150
  x1 <- runif(n, 1, 4)
  x2 <- runif(n, 1, 4)
  m <- 1 + 0.6 * log(x1) + 0.4 * sqrt(x2) # nonlinear frontier
  d <- data.frame(y = m + rnorm(n, 0, 0.25) - abs(rnorm(n, 0, 0.6)),
                 x1 = x1, x2 = x2)

  ## Fan, Li and Weersink, normal-half normal
  f <- npsfm(y ~ x1 + x2, data = d, method = "FLW", dist = "hn")
  f
  head(fitted(f))
  head(f$exp_u_hat)

  ## Simar, Van Keilegom and Zelenyuk: sigma_u and sigma_v vary with x
  g <- npsfm(y ~ x1 + x2, data = d, method = "SVKZ")
  summary(g$sigma.u)
}
```

panel89

*Panel89***Description**

The dataset is a cross-section of U.S. commercial banks for 1989, extracted from the panel dataset used by Kumbhakar, Parmeter and Tsionas (2013) and based on the Federal Reserve Bank of Chicago's Reports of Condition and Income. It contains detailed cost data with inputs and outputs defined under the intermediation approach, and input prices constructed as expense-quantity ratios.

Usage

```
data("panel89")
```

Format

A data frame with 4,985 observations on the following 11 variables.

- y a numeric vector containing the natural logarithm of total cost*
- q1 a numeric vector containing the natural logarithm of installment loans
- q2 a numeric vector containing the natural logarithm of real estate loans
- q3 a numeric vector containing the natural logarithm of business loans
- q4 a numeric vector containing the natural logarithm of federal funds sold and securities purchased
- q5 a numeric vector containing the natural logarithm of other assets
- w1 a numeric vector containing the natural logarithm of the price of labour*
- w2 a numeric vector containing the natural logarithm of the price of capital*
- w3 a numeric vector containing the natural logarithm of the price of purchased funds*
- w4 a numeric vector containing the natural logarithm of the price of interest-bearing deposits in total transaction accounts*
- z a numeric vector containing the natural logarithm of total assets

Details

*The cost and input price variables are normalised by that of a fifth input: the price of interest-bearing deposits in total non-transaction accounts. Total cost is defined as the sum of total expenses for each input. Input prices are derived by dividing the total expense for each input by the corresponding input quantity.

Source

Kumbhakar, S.C., Parmeter, C.F. and Tsionas, E.G. (2013) 'A zero inefficiency stochastic frontier model', *Journal of Econometrics*, 172(1), pp. 66-76. doi:10.1016/j.jeconom.2012.08.021.

References

Kumbhakar, S.C. and Tsionas, E.G. (2005) 'Measuring technical and allocative inefficiency in the translog cost system: a Bayesian approach', *Journal of Econometrics*, 126(2), pp. 355-384. doi:10.1016/j.jeconom.2004.05.006.

Examples

```
data(panel89)
plot(panel89)
```

pcomposed

The distribution of the composed error

Description

Density and distribution function of the stochastic frontier composed error $\varepsilon = v - u$ (production) or $v + u$ (cost), with $v \sim N(0, \sigma_v^2)$ and $u \sim N^+(0, \sigma_u^2)$ independent. The density is a skew-normal and closed form; the distribution function is not, and is obtained here by quadrature that stays accurate far into the tails.

Usage

```
dcomposed(x, sigma_u, sigma_v, inefdec = TRUE, log = FALSE)

pcomposed(q, sigma_u, sigma_v, inefdec = TRUE, lower.tail = TRUE,
          log.p = FALSE, method = c("quadrature", "simulate"),
          n_nodes = 128, R = 1e6, seed = NULL)
```

Arguments

x, q	Numeric vector of quantiles.
sigma_u	Standard deviation of the pre-truncation normal behind u . Zero is allowed and is the no-inefficiency boundary, where the composed error is just the noise.
sigma_v	Standard deviation of the noise, strictly positive.
inefdec	TRUE (the default) for a production frontier, $\varepsilon = v - u$; FALSE for a cost frontier, $\varepsilon = v + u$. The two are mirror images.
log, log.p	Return the log of the density or probability. Use <code>log.p = TRUE</code> whenever the result may be extremely small.
lower.tail	If TRUE (the default) probabilities are $P(\varepsilon \leq q)$, otherwise $P(\varepsilon > q)$. The upper tail is computed directly rather than as $1 - F$; see 'Details'.
method	"quadrature" (the default) is deterministic Gauss-Legendre. "simulate" is the estimator of Amsler, Schmidt and Tsay (2019), kept for comparison.
n_nodes	Quadrature nodes, at least 16.
R, seed	Draws and optional seed for method = "simulate". The seed is restored afterwards, so the caller's random stream is untouched.

Details

Why this exists. Every likelihood in **sfa** evaluates the composed error's density and none needs its distribution function, which is why there was not one. Copula models need it: a Gaussian copula on the composed error contains $\Phi^{-1}(F(\varepsilon))$, so an F that saturates at 0 or 1 does not merely lose precision, it returns an infinity and takes the copula density with it.

How it is computed. Conditioning on u leaves a normal distribution function in closed form, so

$$F(q) = E_u [\Phi((q + su)/\sigma_v)],$$

with $s = +1$ for a production frontier and -1 for a cost frontier. The noise is therefore never drawn or integrated, and the tails inherit the accuracy of **pnorm**, which is reliable twenty standard deviations out. The expectation over u is taken by Gauss-Legendre quadrature on $u = \sigma_u t / (1 - t)$, and accumulated by log-sum-exp so that terms which would underflow individually still contribute.

The upper tail. Requesting `lower.tail = FALSE` negates the argument of Φ rather than forming $1 - F$. At $q = 30$ with $\sigma_u = \sigma_v = 1$ the complement is exactly 1 in double precision while the direct computation returns a finite log-probability below -10^2 .

Accuracy. There is no closed form except at $\lambda = \sigma_u/\sigma_v$ equal to 0 (the normal) or 1, where Amsler, Schmidt and Tsay show $P(Q) = \Phi(Q/(\sqrt{2}\sigma_u))^2$ for the cost case. Against that standard the default quadrature is accurate to about 10^{-13} relative at probabilities as small as 10^{-115} . It is deliberately used everywhere rather than special-cased at $\lambda = 1$: the production form of that identity is a complement, which would reintroduce the saturation the function exists to avoid.

Value

A numeric vector the length of `x` or `q`.

References

Amsler, C., Schmidt, P. and Tsay, W.-J. (2019). Evaluating the cdf of the distribution of the stochastic frontier composed error. *Journal of Productivity Analysis*, 52(1), 29–35.

See Also

[copsfm](#), [sfm](#)

Examples

```
## Density integrates to one.
integrate(function(x) dcomposed(x, 1, 0.5), -Inf, Inf)$value

## Distribution function, and a tail that does not saturate.
pcomposed(c(-2, 0, 2), sigma_u = 1, sigma_v = 0.5)
pcomposed(30, 1, 1, lower.tail = FALSE, log.p = TRUE)

## Checked against the one exact standard there is (cost frontier).
su <- sv <- sqrt(0.5)
c(exact = pnorm(-6 / (sqrt(2) * su))^2,
  computed = exp(pcomposed(-6, su, sv, inefdec = FALSE, log.p = TRUE)))
```

pcomposed_model	<i>Composed-error distribution function for any cross-sectional model</i>
-----------------	---

Description

$P(\varepsilon \leq q)$ for the composed error of any cross-sectional model, where [pcomposed](#) covers only the half-normal case.

Usage

```
pcomposed_model(q, model_name, par, inefdec = TRUE, lower.tail = TRUE,
               log.p = FALSE, n_nodes = 128)

composed_cdf(object, q = NULL, data = NULL, ...)
```

Arguments

q	Numeric vector of quantiles. For composed_cdf, defaults to the fit's own composed-error residuals.
model_name	One of "NHN", "NTN", "NE", "NR", "NU", "NGE", "NLN", "NW", "NG", "NNAK", "TSL", "THT", "tHN".
par	A <i>named</i> parameter vector, as in fit\$out[, "par"].
inefdec	TRUE for a production frontier, FALSE for cost.
lower.tail, log.p	As elsewhere in R.
n_nodes	Quadrature nodes over the inefficiency term.
object	An "sfareg" fit from sfm .
data	The data the model was fitted to. sfm() does not retain it, so it is needed whenever q is not supplied.
...	Passed to pcomposed_model.

Details

The probability is computed as an expectation over the inefficiency term,

$$F(q) = E_u [F_v((q + su)/\sigma_v)],$$

with $s = +1$ for production and -1 for cost. Conditioning on u leaves the noise CDF in closed form, so the noise is never integrated and the tails inherit pnorm()'s (or pt()'s) own accuracy.

Three parameterizations do not read the way their names suggest, and are taken from each likelihood in sfm() rather than from the label: "THT" lists *sigu* *before* *sigv*, the only model that does; "NG"'s *sigu* is the gamma *scale* and its *mu* the *shape*; "NNAK"'s *sigu* is the Nakagami spread, so $\Omega = \text{sigu}^2$; and "NLN"'s *mu* is a *meanlog*.

"THT" **is not an independent convolution**. Tancredi's (2002) composed error is skew- t , a scale mixture in which v and u are divided by the *same* $\sqrt{V/a}$. Treating it as t noise plus an independent

half-normal – which is what "tHN" actually is – gets the log density wrong by up to 1.74, so the two models take different paths here despite looking alike.

Verified two ways: against [pcomposed](#) for the half-normal case, agreeing to 3.5×10^{-15} relative including a lower tail at $\log F = -209$; and for all thirteen models against a four-million-draw simulation, with a maximum absolute error of 4×10^{-4} against a Monte Carlo standard error of 7.5×10^{-4} . Numerically differentiating this CDF reproduces each model's own stored log-density to about 10^{-8} , except for the simulated-ML models "NLN" and "NW", where the difference is the simulation error of their likelihood rather than a disagreement about the model.

Value

A numeric vector of probabilities (or their logs).

See Also

[pcomposed](#), [dcomposed](#), [sfm](#)

Examples

```
p <- c(sigv = 0.5, sigu = 1.1)
pcomposed_model(c(-2, -1, 0, 1), "NE", p)

d <- as.data.frame(data_gen_cs(N = 200, rand = 5, sig_u = 1, sig_v = 0.5,
                             cons = 0.5, beta1 = 0.5, beta2 = 0.5,
                             a = 5, mu = 0.1))
f <- sfm(y_pcs ~ x1 + x2, model_name = "NHN", data = d)
head(composed_cdf(f, data = d))
```

PL80_MVTN

Pitt and Lee (1981) Model III: the multivariate truncated normal panel likelihood

Description

Notes on `psfm(model_name = "PL80_MVTN")`, which fits the likelihood derived in Appendix 2 of Pitt and Lee (1981) but never used there.

Details

The model is

$$y_{it} = x'_{it}\beta + u_{it} + v_{it}, \quad u_{it} \leq 0,$$

with the firm's inefficiency vector $u_i = (u_{i1}, \dots, u_{iT})'$ drawn from a T -variate normal $N(0, \Sigma)$ **truncated to the negative orthant**, and v_{it} iid $N(0, \sigma_v^2)$ independent of u . Unlike `model_name = "PL80"`, which holds inefficiency fixed over time, here it varies across periods *and* is correlated within a firm – Σ carries that dependence.

Why it was never used. Pitt and Lee derived this likelihood and then set it aside, writing that it "is difficult to evaluate since the quantities P_0 and $P(y_i - x_i\beta)$ involve T -dimensional numerical integrals", and estimating Model III by Zellner seemingly-unrelated regression instead. Those

quantities are orthant probabilities of a multivariate normal. `mnormt::sadmvn()` evaluates one in about 3 milliseconds at $T = 6$, so a likelihood evaluation costs roughly $N + 1$ of them – about 0.3 s at $N = 100$, and about 12 s for a whole fit at $N = 80$, $T = 4$. What was intractable in 1981 is merely slow now.

Writing $Q^{-1} = \Sigma^{-1} + I/\sigma_v^2$ and $\mu_i = Q\varepsilon_i/\sigma_v^2$, the per-firm log density is

$$\log f(\varepsilon_i) = -\frac{T}{2} \log 2\pi - T \log \sigma_v - \frac{1}{2} \log |\Sigma| + \frac{1}{2} \log |Q| - \log P_0 - \frac{1}{2} \left(\frac{\varepsilon_i' \varepsilon_i}{\sigma_v^2} - \frac{\varepsilon_i' Q \varepsilon_i}{\sigma_v^4} \right) + \log P_i,$$

where $P_0 = \Pr(w \leq 0)$ for $w \sim N(0, \Sigma)$ is the truncation constant and $P_i = \Pr(w \leq 0)$ for $w \sim N(\mu_i, Q)$.

Σ is equicorrelated, not unrestricted. It is parameterized as $\Sigma = \sigma_u^2[(1 - \rho)I + \rho \mathbf{1}\mathbf{1}']$, costing two parameters. An unrestricted Σ costs $T(T + 1)/2 - 21$ at $T = 6$, 55 at $T = 10$ – on top of β and σ_v , every one identified only through orthant probabilities. The equicorrelated form captures what the general Σ was introduced for: dependence of a firm's inefficiency across periods. $\rho = 0$ gives inefficiency independent over time; $\rho \rightarrow 1$ approaches the time-invariant "PL80" case. Because the form is equicorrelated every matrix quantity above is closed form (Sherman-Morrison), so only the orthant probabilities are numerical.

Requirements and limits. A **balanced** panel with $T \geq 2$: Σ is a single $T \times T$ matrix shared by every firm, and with $T = 1$ there is no cross-period dependence for it to describe. An unbalanced panel is an error pointing at "PL80". Reported parameters are `sigv`, `sigu`, `rho` and the frontier coefficients.

`u_hat` is the posterior mean of u from the Gaussian part, floored at zero, **not** $E[u_{it} | \varepsilon_i]$: the exact conditional mean of a truncated multivariate normal is another T -dimensional integral. It is exact where the truncation does not bind and an approximation where it does.

Author(s)

David H. Bernstein

References

Pitt, M.M. and Lee, L.-F. (1981) 'The measurement and sources of technical inefficiency in the Indonesian weaving industry', *Journal of Development Economics*, 9(1), pp. 43-64. See Appendix 2 for the likelihood.

See Also

[psfm](#), [data_gen_p](#) for the `y_pl_mvtn` column that tests it.

Examples

```
library(sfa)
d <- data_gen_p(t = 4, N = 60, rand = 5, sig_u = 1, sig_v = 0.3, sig_r = 0.2,
               sig_h = 0.4, cons = 0.5, beta1 = 0.5, beta2 = 0.5)
f <- psfm(y_pl_mvtn ~ x1 + x2, model_name = "PL80_MVTN",
          data = as.data.frame(d), individual = "name")
f$out
```

predict.sfareg	<i>Predictions, fitted values and residuals for stochastic frontier fits</i>
----------------	--

Description

Standard modelling generics for objects of class "sfareg", as returned by [sfm](#), [psfm](#), [zsfm](#) and [ttsfm](#).

Usage

```
## S3 method for class 'sfareg'
predict(object, newdata = NULL,
        type = c("frontier", "response", "efficiency"), ...)
## S3 method for class 'sfareg'
fitted(object, ...)
## S3 method for class 'sfareg'
residuals(object, ...)
```

Arguments

object	An object of class "sfareg".
newdata	Optional data frame containing the frontier regressors at which to predict. If omitted, the data the model was fitted to is used.
type	"frontier" (default) returns the deterministic frontier $x'\beta$. "response" shifts it by the predicted inefficiency, giving an estimate of $E[y \mid x]$ rather than of the frontier itself, with the sign following the production/cost convention the model was fitted under. "efficiency" returns the predicted technical efficiency $E[\exp(-u) \mid \varepsilon]$.
...	Ignored.

Details

Only the first part of a pipe formula ($y \sim x \mid z \mid zp$) describes the frontier; later parts parameterize variances and take no part in prediction.

Frontier coefficients are identified by matching the fitted coefficient names against the frontier design matrix, so these methods work for every model in the package without depending on where a given model happens to place its auxiliary parameters (lambda/sigma for NHN, sigv/sigu for NE, sigmaSq/gamma for PL80, and so on).

type = "efficiency" and type = "response" are available only for the estimation sample, since both require the composed residual and therefore the response; requesting them with newdata raises an error rather than returning something misleading. "efficiency" additionally requires a model that returns an efficiency prediction – see [sfm](#) for which ones do.

Value

A numeric vector, one element per observation.

Author(s)

David H. Bernstein

See Also[sfm](#), [psfm](#), [sfareg-methods](#)**Examples**

```
library(sfa)
d <- data_gen_cs(N = 200, rand = 1, sig_u = 0.8, sig_v = 0.3,
                 cons = 1, beta1 = 0.5, beta2 = 0.5, a = 5, mu = 0.5)
f <- sfm(y_pcs ~ x1 + x2, model_name = "NHN", data = d)

head(fitted(f))
head(residuals(f))
head(predict(f, type = "efficiency"))
head(predict(f, newdata = d[1:5, ]))
```

print.sfareg

Print method for "sfareg" objects

Description

Compact display of a stochastic frontier fit, as returned by [sfm](#), [psfm](#), [zsfm](#) or [ttsfm](#).

Usage

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

Arguments

x An object of class "sfareg", as returned by [sfm](#), [psfm](#), [zsfm](#) or [ttsfm](#).
... Additional arguments passed to other methods

Details

Prints the model name, the estimated coefficients with their standard errors and t-values (the out matrix), and the log-likelihood where the estimator has one. For a fuller display, including the frontier and variance-parameter blocks, use [summary.sfareg](#).

Value

No return value, called for side effects

Author(s)

David H. Bernstein

See Also[summary.sfareg](#), [sfareg-methods](#)**Examples**

```
library(sfa)

cs_data_trial <- data_gen_cs(N= 1000, rand = 1, sig_u = 0.3, sig_v = 0.3,
  cons = 0.5, beta1 = 0.5, beta2 = 0.5, a = 4, mu = 1)

cs.nhnz <- sfm(formula = y_pcs_z ~ x1 +x2| z, model_name = "NHN_Z",
  data = cs_data_trial, PSopt = TRUE)

print(cs.nhnz)
```

psfm

*psfm***Description**

Function to implement various panel data stochastic frontier estimators

Usage

```
psfm(formula, model_name = c("TRE_Z", "GTRE_Z", "TRE",
  "GTRE", "GTRE_FML", "TFE", "TFE_WMLE", "FD",
  "GTRE_SEQ1", "GTRE_SEQ2", "SSFE",
  "PL80", "PL80_MVTN", "BC92", "K1990", "K1990modified",
  "CSS", "LS", "SSRE", "SSCRE", "KSS"), data,
  maxit.bobyqa = 5000, maxit.nlminb = 500,
  maxit.psoptim = 100, maxit.optim = 1000,
  REPORT = 1, trace = 3, pgtol = 0, individual,
  halton_num = NULL, start_val = FALSE, gamma = FALSE,
  PSopt = FALSE, optHessian = TRUE, inefdec = TRUE, Method = "L-BFGS-B",
  verbose = FALSE, rand.gtre = NULL, rand.psoptim = NULL,
  OPG_calc = FALSE,
  estimator = c("fiml", "sml", "seq1", "seq2"),
  collinear_action = c("start_only", "error", "warn_drop"),
  time = NULL, tfe_lambda_max = 100,
  keep_objective = FALSE, kss_L = "auto",
  kss_smooth = "auto", kss_L_max = 7L, mundlak = NULL)
```

Arguments

formula	a symbolic description for the model to be estimated
model_name	Which panel estimator to fit. All are described in ‘Details’; in brief: "TRE", "TRE_Z" Greene’s (2005) true random effects, by simulated maximum likelihood. <code>_Z</code> adds determinants of σ_u. "GTRE", "GTRE_Z" the four-component model separating persistent from transient inefficiency, by simulated maximum likelihood. <code>_Z</code> adds determinants. "GTRE_FML" the same four-component model by full information maximum likelihood through its closed-skew-normal representation; deterministic, but requires a balanced panel. "GTRE_SEQ1", "GTRE_SEQ2" two-step/moment-based estimators of the four-component model. Not maximum likelihood – no <code>logLik()</code>. "TFE" Greene’s (2005) true fixed effects. Changed meaning in 1.1.3 – see ‘Details’. "TFE_WMLE" Chen, Schmidt and Wang’s (2014) within maximum-likelihood estimator. "FD" the first-difference estimator of Wang and Ho (2010). "SSFE" Schmidt and Sickles’ (1984) fixed-effects estimator, inefficiency read off the firm effects. Not maximum likelihood – no <code>logLik()</code>. "CSS" Cornwell, Schmidt and Sickles (1990): a firm-specific quadratic in time, so inefficiency varies over time and firms may cross over. Not maximum likelihood – no <code>logLik()</code>. "LS" Lee and Schmidt (1993): one common temporal pattern scaled per firm, $\alpha_{it} = \delta_t \alpha_i$. Time-varying but no crossover. Not maximum likelihood – no <code>logLik()</code>. "SSRE" Schmidt and Sickles (1984) random effects: the GLS estimator, with inefficiency read off the firm effects’ BLUPs. More efficient than "SSFE" and identifies time-invariant regressors, at the cost of assuming the effects are uncorrelated with the regressors. Not maximum likelihood – no <code>logLik()</code>. "SSCRE" Correlated random effects: "SSRE" augmented with the within-firm means of the time-varying regressors (Mundlak 1978), so that correlation between effects and regressors is modelled rather than assumed away. Not maximum likelihood – no <code>logLik()</code>. "KSS" Kneip, Sickles and Song (2012): the firm effect is a smooth function of time lying in an L-dimensional space whose basis is estimated from the data. Nests "SSFE", "LS" and "CSS". Balanced panels only. Not maximum likelihood – no <code>logLik()</code>. "PL80" Pitt and Lee (1980): time-invariant inefficiency. "BC92" Battese and Coelli (1992): exponential time decay. "PL80_MVTN" Pitt and Lee’s (1981) Model III: inefficiency varies over time and is CORRELATED within a firm, through a multivariate normal truncated to the negative orthant. This is the likelihood from their Appendix 2, which they derived but did not use, calling the T-dimensional integrals intractable. Balanced panels only. See PL80_MVTN. "K1990", "K1990modified" Kumbhakar (1990) time patterns for inefficiency.

	Matching ignores case, so "nhn", "NHN" and "Nhn" are the same choice.
data	A panel data set. This may be an ordinary <code>data.frame</code> (or <code>tibble/data.table</code>) – the panel index is constructed internally from the individual and time arguments – or a <code>plm::pdata.frame</code> that already carries an index, which is used as-is.
maxit.bobyqa	Maximum number of iterations for the bobyqa optimization routine
maxit.nlmnb	Maximum number of iterations for the nlmnb optimization routine.
maxit.psoptim	Maximum number of iterations for the psoptim optimization routine
maxit.optim	Maximum number of iterations for the optim optimization routine
REPORT	reporting parameter
trace	Integer. Tracing level passed through to the optimizer; larger values print more.
pgtol	Numeric. Projected-gradient tolerance passed to <code>optim()</code> 's "L-BFGS-B" method. 0 uses the optimizer's own default.
individual	individual unit in the regression model
halton_num	number of Halton draws to use in SML models
start_val	starting value (optional)
gamma	Logical. Report the variance parameters in the $\gamma = \sigma_u^2 / (\sigma_u^2 + \sigma_v^2)$ parameterization instead of the default. Defaults to FALSE.
PSopt	use psoptim optimization routine (T or F)
optHessian	Logical. Should a numerically differentiated Hessian matrix be returned while using the optim routine? (for optim routine)
inefdec	Production or cost function
Method	The method to be used for optim. See 'Details' within optim.
verbose	Logical. Print optimization progress messages? Default is FALSE.
rand.psoptim	Integer. Seed for replication of psoptim. Default to NULL.
rand.gtre	Integer. Seed for replication of the gtre model. Default to NULL.
OPG_calc	Logical. Should the outer product of gradients be calculated for inference. Default to FALSE.
estimator	How to estimate the four-component GTRE model, in the same spirit as <code>sfm</code> 's estimator: one model, several routes to it. Applies to <code>model_name = "GTRE"</code> only, and is ignored with a warning elsewhere. "fm1" the default. Full information maximum likelihood through the closed-skew-normal representation. Deterministic – nothing is integrated by Monte Carlo, so repeated fits to the same data agree exactly. Requires a balanced panel. "sml" Simulated maximum likelihood over Halton draws. Handles unbalanced panels. This is what <code>model_name = "GTRE"</code> meant through version 1.1.3. "seq1", "seq2" The two-step moment-based estimators. Not maximum likelihood, so they carry no <code>logLik()</code> .

The older names remain and are unchanged: "GTRE_FML", "GTRE_SEQ1" and "GTRE_SEQ2" select the same three routes directly.

On an **unbalanced** panel "fml" cannot be fitted. If you simply took the default, psfm() warns and falls back to "sml" – erroring would make "GTRE" unusable by default on a whole class of data. If you asked for "fml" explicitly it errors instead, rather than silently fitting a different estimator than the one you named.

collinear_action	One of "start_only" (default), "error", or "warn_drop". Controls what happens when the <i>between-individual</i> design matrix used to build starting values is rank deficient. This is checked separately from the pooled design because the two can differ: time dummies, for example, are routinely estimable in a pooled specification while collapsing onto the intercept once averaged within each unit, which is exactly the situation that makes the internal random-effects initialization fail. "start_only" keeps the model you asked for and removes the offending columns from the starting-value regression only, taking their starting values from a pooled OLS fit; "error" stops before estimation and names the offending columns; "warn_drop" removes those terms from the estimated model. All three report what was found. Only relevant to the models initialized by a random-effects regression (GTRE, TRE and their _Z/SEQ variants).
time	Character. Name of a time-period column in data, used by the error-components frontier models ("PL80", "BC92", "K1990", "K1990modified"). If NULL (the default), a sequential within-individual time index is constructed from row order, which assumes data is already sorted chronologically within each individual.
tfe_lambda_max	Numeric. Upper bound placed on $\lambda = \sigma_u/\sigma_v$ when fitting model_name = "TFE"; ignored by every other model. Defaults to 100, i.e. σ_v is not allowed below about 1% of σ . This is a guard against a genuine degeneracy of Greene's likelihood rather than a convergence tolerance – see 'Details'. A fit that pins at the bound warns and should not be read as an estimate of λ .
keep_objective	Retain the likelihood closure on the fitted object as \$objective, for "GTRE", "TRE", "GTRE_Z" and "TRE_Z" – the simulated-ML panel models. This lets the log-likelihood be evaluated away from the optimum, which is what profiling a single parameter requires; sfa_diagnostics uses it, and it is how the sigh convergence question was localized (see 'Details'). FALSE by default, since the closure captures the data and the draws and so is not small.
kss_L	"KSS" only. The dimension L of the factor space. "auto" (the default) selects it by the Bai and Ng (2002) IC_{p2} criterion; give an integer to fix it.
kss_smooth	"KSS" only. The smoothing penalty applied to each firm's residual trajectory before the basis is extracted. "auto" (the default) chooses it by generalized cross-validation; 0 turns smoothing off, which reduces the model to an unrestricted factor model.
kss_L_max	"KSS" only. Largest L the automatic criterion may choose. The criterion is capped internally at $\lfloor T/2 \rfloor$ as well, for the reason given in 'Details', and warns if it selects that cap. An explicit kss_L is not subject to the $\lfloor T/2 \rfloor$ cap, only to $\min(T - 1, n - 1)$.
mundlak	Mundlak adjustment terms: a formula naming the regressors whose within-firm means should be added to the frontier, or ~ . for every frontier regressor.

NULL (the default) adds nothing. Only for models that treat the firm effect as random; "TFE", "TFE_WMLE", "FD" and "SSFE" refuse it, because they have already removed the effect. See 'Details'.

Details

The generalized true random effects model (GTRE, 4-component model) and true random effects models (TRE) are both estimated by simulated maximum likelihood based on the paper by the Fillipini and Greene (2016, JPA). The TRE_Z and GTRE_Z allow for modeling the u-component of the GTRE and TRE with determinants of inefficiency. The first-difference estimator (FD) of Wang and Ho (2010, JoE) is also available.

Choosing how GTRE is estimated. Since version 1.1.4 the four routes to the four-component model are selected with `estimator` rather than by picking among four `model_name` values that looked like four different models. `model_name = "GTRE"` now defaults to `estimator = "fml"`; through 1.1.3 it fitted the simulated-ML estimator, which is `estimator = "sml"`. Scripts that pass "GTRE" therefore get a different estimator than they did and are warned once per call. The explicit names "GTRE_FML", "GTRE_SEQ1" and "GTRE_SEQ2" are unaffected.

Two ways of estimating the same four-component model. "GTRE" and "GTRE_FML" both fit

$$y_{it} = \beta_0 + x'_{it}\beta + r_i - h_i + v_{it} - u_{it},$$

with r_i, v_{it} two-sided and h_i, u_{it} one-sided (half-normal), separating *persistent* inefficiency h_i from *transient* inefficiency u_{it} . They differ in how the likelihood is obtained.

"GTRE" uses simulated maximum likelihood over Halton draws. "GTRE_FML" uses *full information* maximum likelihood through the model's closed-skew-normal representation: with $A = -[1_T, I_T]$, $V = \text{diag}(\sigma_h^2, \sigma_u^2 I_T)$ and $\Sigma = \sigma_v^2 I_T + \sigma_r^2 1_T 1_T'$, the firm-level density is available in closed form and nothing is integrated by Monte Carlo. The likelihood is therefore deterministic, and repeated fits to the same data give identical answers.

The $(T+1)$ -dimensional normal CDF this requires would normally be the bottleneck, but its covariance is diagonal-plus-rank-one for this model, so it collapses to a one-dimensional Gauss-Hermite integral – the same reduction used for "TFE".

"GTRE_FML" reports **raw standard deviations** `sigr`, `sigv`, `sigh`, `sigu` (individual effect, noise, persistent inefficiency, transient inefficiency), *not* the λ/σ reparameterization "GTRE" uses, and it requires a **balanced** panel – the closed-skew-normal pieces are built at a single T . It errors rather than silently mis-estimating if the panel is unbalanced; use "GTRE" in that case.

Starting values for "GTRE_FML". The closed-skew-normal likelihood carries a boundary optimum at $\sigma_h = 0$, where the model collapses to "TRE" and the intercept absorbs the missing $E[h] = \sigma_h \sqrt{2/\pi}$. *Two different things can put a fit there, and the likelihood tells them apart.* A fit whose boundary solution has a *lower* likelihood than the true parameter vector reached it by getting stuck, and is an optimizer failure; one whose boundary solution has a *higher* likelihood reached it because that is where the maximum is, and is the correct answer for that sample (see 'When a persistent scale collapses to zero' below). The start search addresses only the first kind. Following Colombi (2010) and Colombi, Martini and Vittadini (2011), who recommend seeding this likelihood from the two-step moment estimates, `psfm()` builds two candidate starts – the random-effects decomposition shared with "GTRE", and the two-step estimator reported by "GTRE_SEQ2" – evaluates the likelihood at each, and begins from the better one. Which was chosen, and the log-likelihood at both, is returned in `start_search`. Supplying `start_val` overrides the search.

Note that σ_r and σ_h enter the density only through $\sigma_r^2 + \sigma_h^2$; their separation rests entirely on the skewness term. That pair is therefore the least precisely estimated part of the model even when the fit is at the global optimum, which is consistent with the confounding between latent heterogeneity and persistent inefficiency reported in the source papers.

"GTRE_SEQ1" and "GTRE_SEQ2" are large-T estimators. Both take the random-effects decomposition from plm and hand its two output series to a second stage – "GTRE_SEQ1" fits an intercept-only normal-half-normal likelihood to each, "GTRE_SEQ2" inverts their second and third moments. The second stage treats those series as if they were draws from $v_{it} - u_{it}$ and $r_i - h_i$, and at finite T they are not: α_hat is the shrunk predictor $c(\alpha_i + \bar{e}_i)$ with $c = T\sigma_\alpha^2 / (T\sigma_\alpha^2 + \sigma_\epsilon^2)$, and ϵ_hat is the quasi-demeaned residual $(1 - \theta)\alpha_i + e_{it} - \theta\bar{e}_i$ with $\theta = 1 - \sigma_\epsilon / \sqrt{T\sigma_\alpha^2 + \sigma_\epsilon^2}$.

Relative to the latent draws this attenuates the third central moment on the ϵ side by $(1 - \theta/T)^3 + (T - 1)(-\theta/T)^3$ (about 0.81 at $T = 10$) and scales the α side by c^2 and c^3 . None of these factors tends to 1 as N grows with T fixed, so **both estimators converge to the wrong constants and are inconsistent for fixed T** ; each call warns accordingly. On the package's own convergence design ($T = 10$, $\sigma_h = 0.8$, $\sigma_r = 0.4$) the estimates settle at $\gamma_{uv} \approx 0.87$ against 0.917 and $\sigma_{hr}^2 \approx 0.68$ against 0.80, and the log MSE-on-log N slopes are near zero rather than -1 . The same routines applied to the latent draws themselves recover the truth at the $\sqrt{N}$ rate, which locates the problem in the first-stage decomposition rather than in either second stage. Raising T removes the bias; raising N only sharpens it. For a consistent alternative use `model_name = "GTRE"`, which estimates all four variance components jointly.

The standard errors these two report are delta-method errors built from the *iid* moment variances, so they describe sampling variability around each estimator's probability limit, not around the truth. They are also optimistic on the ϵ side, whose n residuals are correlated within firm.

The two true-fixed-effects estimators, and a change of meaning in "TFE". Both "TFE" and "TFE_WMLE" fit the same model,

$$y_{it} = \alpha_i + x'_{it}\beta + v_{it} - u_{it}, \quad v \sim N(0, \sigma_v^2), \quad u \sim N^+(0, \sigma_u^2),$$

with α_i an unrestricted individual effect; they differ in how they estimate it.

"TFE" is Greene's (2005, JoE/JPA) true fixed effects estimator: the ordinary composed-error likelihood with the N individual intercepts estimated jointly with (λ, σ, β) . This package concentrates the α_i out at every likelihood evaluation (each is the unique root of a strictly decreasing score, solved by safeguarded Newton to machine precision), so the reported parameter vector, its standard errors and the out layout are the same $2 + K$ as "TFE_WMLE". It is the estimator subject to the *incidental parameters problem*: with T fixed, σ_u is biased upward and efficiency scores correspondingly pessimistic.

"TFE_WMLE" is Chen, Schmidt and Wang's (2014, JoE) within maximum-likelihood estimator, which applies maximum likelihood to the deviations from individual means. The α_i drop out of the likelihood entirely, so the estimator is free of the incidental parameters problem; the firm effects are recovered afterwards from the mean residual with the usual $\sqrt{2/\pi} \sigma_u$ adjustment.

The two are *not* nested and their log-likelihoods are not on the same scale – "TFE" evaluates the likelihood of the data, "TFE_WMLE" that of the within-transformed deviations – so `logLik()`, `AIC()` and `BIC()` must not be used to choose between them.

Through `sfa 1.1.2`, `model_name = "TFE"` selected the Chen-Schmidt-Wang estimator. It now selects Greene's, matching standard usage; the Chen-Schmidt-Wang estimator is "TFE_WMLE". Existing scripts that pass "TFE" therefore get a different estimator than they did, and are warned once per call.

Degeneracy of the Greene likelihood. Because α_i is unrestricted, Greene's likelihood always has a supremum on the $\sigma_v \rightarrow 0$ boundary: setting $\alpha_i = \max_t(y_{it} - x'_{it}\beta)$ makes every composed error weakly negative, $\Phi(-\lambda e/\sigma) \rightarrow 1$, and what remains is the deterministic-frontier likelihood, which is finite and attained. On simulated panels this supremum is reached along a path that first dips below the interior local maximum and then rises past it, so an unconstrained search that escapes the interior basin returns $\sigma_v \approx 0$ together with meaningless standard errors; when λ is large and T small, no interior maximum need exist at all. `tfe_lambda_max` bounds the search accordingly, and a fit that pins at the bound warns. This is a property of the estimator, not of the implementation, and is one of the motivations for "TFE_WMLE".

Both problems are small- T phenomena. Simulations in which T grows with N fixed at 100 (truth $\lambda = 3.333$) recover $\hat{\lambda} = 88.5, 4.8, 3.6, 3.6, 3.5$ at $T = 10, 20, 30, 40, 50$: the fit is pinned at the bound at $T = 10$ and has converged by $T = 50$. The degeneracy and the incidental-parameters bias dissolve together as T grows, so `tfe_lambda_max` matters on short panels and is irrelevant on long ones. "TFE_WMLE", having no incidental parameters, is unaffected by T either way.

"SSFE" is the classical Schmidt and Sickles (1984, JBES) fixed-effects (LSDV) estimator: a standard within/fixed-effects panel regression, with technical inefficiency read directly off the estimated firm effects (each firm's distance from the best-performing firm in the sample) rather than estimated by maximum likelihood. It requires no numerical optimization and returns no log-likelihood – `logLik()/AIC()/BIC()` are not defined for it (see [sfareg-methods](#)). Note this is a different model from "TFE"/"TFE_WMLE" above, despite the similar names: both of those are maximum-likelihood estimators that allow a separate stochastic noise term on top of the fixed effects, while "SSFE" attributes all within-firm deviation to inefficiency directly.

"K1990" and "K1990modified" (Kumbhakar, 1990) extend the same error-components frontier with richer time paths for inefficiency: "K1990" uses $B_{it} = (1 + \exp(bt + ct^2))^{-1}$ and "K1990modified" uses $B_{it} = 1 + d(t - T_i) + e(t - T_i)^2$, against "BC92"'s $\exp(-\eta(t - T_{ref}))$ and "PL80"'s time-invariant $B_{it} = 1$. All four share one likelihood, differing only in B_{it} . Note that "K1990"'s b and c are weakly identified – quite different pairs produce near-identical fitted time paths – so the fitted B_{it} path is more interpretable than either coefficient on its own.

"PL80" (time-invariant panel inefficiency, Pitt and Lee, 1980, JoE) and "BC92" (time-varying decay, Battese and Coelli, 1992, JPA) are both natively estimated maximum-likelihood implementations of the same "error components frontier": $y_{it} = x_{it}'\beta + v_{it} - B_{it}u_i$ (production; sign flips for cost), with v_{it} iid normal, u_i iid half-normal and constant across time within a firm, and $B_{it} = 1$ for "PL80" or $B_{it} = \exp(-\eta(t - T_{ref}))$ for "BC92" (T_{ref} being the last time period in the whole panel). The closed-form log-likelihood is obtained by integrating u_i out of the joint density of (ϵ_{it}, u_i) ; it nests the ordinary cross-sectional normal-half-normal likelihood when every firm has a single observation. Verified against `frontier::sfa()` (matching coefficients, log-likelihood, and `frontier::efficiencies()`-style predicted efficiencies to numerical precision across balanced/unbalanced panels and production/cost specifications) before that dependency was removed. The time argument above supplies each observation's period for "BC92"'s decay term; if left NULL, a sequential within-individual time index is constructed from row order, which assumes data is already sorted chronologically within each individual.

Note on default iteration counts: `psfm()`'s defaults were raised in version 1.1.3, from 100/10/10 for `maxit.bobyqa/maxit.psoptim/maxit.optim` to 5000/100/1000, and `maxit.nlminb` (previously hard-coded) is now an argument. The old values were binding rather than merely economical: the K1990/K1990modified branch already carried a note that 100 bobyqa evaluations left its seven-parameter fits several log-likelihood units short of the optimum purely on the iteration cap, and GTRE_FML at $N = 500, T = 10$ roughly halves its root-mean-square error against known true values

when the caps are lifted, at about 1.5 times the run time. They remain below `sfm()`'s defaults because several `psfm()` models (GTRE, GTRE_Z, TRE, TRE_Z) are estimated by simulated maximum likelihood over Halton draws, so each objective evaluation is far more expensive than the closed-form likelihoods used elsewhere. Lower them explicitly if a fast approximate call is wanted.

Value

An object of class "sfareg" containing components that vary by model. All models return:

<code>out</code>	A matrix with parameter estimates, standard errors, and t-values.
<code>opt</code>	A list containing the optimization results from the final optimization procedure (not returned for GTRE_SEQ1 and GTRE_SEQ2).
<code>total_time</code>	The total computation time for model estimation.
<code>start_v</code>	The starting values used in the optimization (not returned for GTRE_SEQ1 and GTRE_SEQ2).
<code>model_name</code>	The name of the panel stochastic frontier model estimated.
<code>formula</code>	The formula used in the model specification.
<code>coefficients</code>	A vector of estimated parameters.
<code>std.errors</code>	A vector of standard errors for the estimated parameters (NA if <code>optHessian = FALSE</code>).
<code>t.values</code>	A vector of t-values for the estimated parameters (NA if <code>optHessian = FALSE</code>).
<code>call</code>	The matched call.
<code>data</code>	The data used in estimation.

Additional model-specific components:

For GTRE_FML model:

<code>start_search</code>	A list recording the starting-value search: <code>n_tried</code> , the log-likelihood at each candidate (<code>loglik</code>), and which was chosen ("random-effects" or "two-step"). NULL when <code>start_val</code> was supplied.
<code>U</code>	Predicted transient (time-varying) technical efficiency for each observation.
<code>H</code>	Predicted persistent (time-invariant) technical efficiency for each individual, named by individual.

For GTRE and GTRE_Z models:

<code>H</code>	Predicted time-invariant technical efficiency for each individual.
----------------	--

For GTRE_Z model:

<code>out_opg</code>	OPG (outer product of gradients) standard errors and t-values along with parameters.
<code>out_sandwich</code>	sandwich standard errors and t-values along with parameters.

For GTRE, GTRE_Z, TRE and TRE_Z models:

<code>U</code>	Predicted time-varying technical efficiency for each observation.
----------------	---

For TFE and TFE_WMLE models:

r_hat_m	Estimated individual-specific fixed effects, named by individual. For "TFE" these are the maximum-likelihood estimates of α_i directly; for "TFE_WMLE" they are recovered after estimation from the mean residual plus $\sqrt{2/\pi} \sigma_u$, since the within likelihood does not identify them.
exp_u_hat	Predicted technical efficiency, $E[\exp(-u) \epsilon]$ (Battese and Coelli, 1988).

For TFE model only:

u_hat	Predicted inefficiency, $E[u \epsilon]$ (Jondrow et al., 1982).
-------	---

For FD model:

u_hat	Predicted technical efficiency in levels.
h_hat	Estimated z heterogeneity function values.
exp_u_hat	Predicted technical efficiency.

For GTRE_SEQ1 and GTRE_SEQ2 models:

other_parms	A matrix of additional parameters (lambda, sigma, beta_0 for SEQ1; sigma_u, sigma_v, sigma_h, sigma_r, lambda, sigma for SEQ2).
-------------	---

When a persistent scale collapses to zero. GTRE carries two persistent components – σ_h , persistent inefficiency, and σ_r , the firm random effect – and the likelihood cannot always separate them in a given sample. When it cannot, it merges them: one scale goes to zero and the other absorbs its variation, coming back inflated.

This is common rather than exceptional. On simulated data at $T = 6$ with both components genuinely present ($\sigma_r = 0.20$, $\sigma_h = 0.40$), one of the two collapsed in **37% of 87 replications**, and the commoner direction was $\sigma_r \rightarrow 0$ rather than $\sigma_h \rightarrow 0$. The share with both interior does improve with N (15, 19 and 21 of about 29 at $N = 50, 100, 200$) but slowly.

A collapse is usually the right answer, not a numerical failure. On one such replication – whose realized σ_h scale was 0.3695, so persistent inefficiency was certainly present – the boundary solution had a log-likelihood *higher than the true parameter vector* by 3.66, a likelihood ratio of about 39. This is the panel counterpart of the cross-sectional wrong-skewness result (Waldman 1982), where $\sigma_u = 0$ is the correct maximum likelihood estimate rather than an optimizer failure; see [skewness_test](#). This is not in tension with the start search described above under "GTRE_FML": a boundary fit that beats the truth on likelihood is the estimate, while one that loses to it is a trapped optimizer, and the search exists to remove the second so that what remains is the first.

What a collapse does to the rest of the fit, exactly. One rule covers it: *a collapse conserves the persistent variance and relocates the persistent mean into the intercept*. Those are two different destinations. h is half-normal, so it carries both a mean $E[h] = \sigma_h \sqrt{2/\pi}$ and a variance $\text{Var}(h) = \sigma_h^2(1 - 2/\pi)$, and the model has a separate place for each: the frontier intercept takes the mean, since $E[y] = \beta_0 - E[h] - E[u]$, and the surviving persistent scale takes the variance. The intuitive shortcut – that the survivor comes back at $\sqrt{\sigma_h^2 + \sigma_r^2}$ – ignores that split and is wrong.

	surviving scale	intercept
$\sigma_h \rightarrow 0$	$\hat{\sigma}_r = \sqrt{\sigma_r^2 + \sigma_h^2(1 - 2/\pi)}$	shifts by $-E[h]$
$\sigma_r \rightarrow 0$	$\hat{\sigma}_h = \sqrt{\sigma_h^2 + \sigma_r^2/(1 - 2/\pi)}$	shifts by $+\Delta E[h]$

At $\sigma_h = 0.40$, $\sigma_r = 0.20$ that predicts 0.3133 and -0.3192 in the first row, 0.5197 and $+0.0955$ in the second, with nothing fitted. Over 1000 replications at each of five sample sizes the measurements are 0.2973, -0.3035 , 0.5058 and $+0.0931$ – within 2.5% to 5.1%, and all four slightly below prediction, which is what selection produces rather than a wrong constant.

The practical consequence is that a collapsed fit does not merely mislabel the persistent split. It displaces the intercept by a known and substantial amount, so any level read off `coef()` is displaced with it.

What a collapse is telling you about your data. It is wrong skewness, measured between firms rather than within them. Take the OLS residuals, average them within each firm – that average is the persistent part of the error – and look at their skew. Over 1000 replications, the samples whose between-firm skew came out positive (the wrong way for a production frontier) lost σ_h to the boundary **69%** of the time, against 3.6% for the rest; the area under the ROC curve for that single statistic is **0.955**. Not one wrong-skew sample collapsed σ_r instead. So a collapsed σ_h is not a numerical accident to be retried with different starting values: this sample carries no between-firm evidence of persistent inefficiency, and the estimate is reporting that.

Note that this is a reading of the sign, not a test. The firm mean of the residuals contains $-\bar{u}_i$ as well as $-h_i$, and the mean of T half-normals is itself skewed the same way, so on short panels the between-firm skew leans negative even when $\sigma_h = 0$ – 82% of the time at $T = 4$, 63% at $T = 10$, 53% at $T = 30$. `skewness_test` is therefore *not* applicable to the between-firm residuals as it stands.

Whether firms or periods help depends on the estimator, and the two answers are opposite. Crossed (N, T) designs on the same data-generating process, fitting the collapse rate on $\log N$ and $\log T$:

estimator	fits	log N	log T
"fiml" (the default)	600	-1.66 (0.18)	-0.93 (0.20)
"sml"	251	-0.68 (0.26)	-1.07 (0.27)

Under the default, firms buy about 1.8 times what periods do; under simulated ML it is the other way round, periods buying about 1.6 times what firms do. Restricting the "fiml" design to $N \leq 200$, cell for cell the same grid the "sml" one used, leaves it at -1.64 (0.22), so this is the estimator and not the design; the difference in the N coefficient is significant ($z = 2.8$) and the difference in the T coefficient is not.

The practical reading, for the estimator you get by default: **add firms**. At $T = 4$ the collapse rate falls 68%, 36%, 18%, 6% across $N = 50, 100, 200, 400$, and reaches 0% at $N = 400, T = 16$. Under "sml" the same row is flat – 60%, 39%, 41% – which is why a short panel really is a dead end there however many firms it has. If you must use estimator = "sml" (an unbalanced panel forces it), then length is what buys the split, and a four-period panel will not give it to you.

How to tell in advance whether your panel can support the split. One quantity governs it, and it is not σ_h/σ_r . What the likelihood has to do is read the persistent part out of firm means that the transient block contaminates, so write the firm-mean composite as $r_i - h_i - \bar{u}_i + \bar{v}_i$ and compare its persistent variance with the transient variance left in it:

$$S/N = \frac{\sigma_r^2 + \sigma_h^2(1 - 2/\pi)}{[\sigma_u^2(1 - 2/\pi) + \sigma_v^2]/T}$$

Across six designs varying all four scales in different combinations (50 replications each, $N = 100$, $T = 6$), this orders the collapse rate **perfectly** – Spearman -1.000 under both estimators, and a logistic coefficient on $\log(S/N)$ of -1.15 (0.20):

S/N	collapse	$\sigma_h \rightarrow 0$	$\sigma_r \rightarrow 0$
0.67	54%	0.14	0.40
0.90	50%	0.10	0.40
0.96	40%	0.21	0.19
1.30	31%	0.15	0.17
5.20	8%	0.06	0.02
11.69	6%	0.04	0.02

As a rule of thumb: below $S/N \approx 1$ about half your fits will merge the two components, and it takes roughly $S/N > 5$ to get under 10%.

Two things follow that are easy to get backwards. First, **raising** σ_h/σ_r **does not help** – it only decides which scale dies. Going from a ratio of 2 to 4 by shrinking σ_r takes $\sigma_h \rightarrow 0$ down (0.15 to 0.10) and $\sigma_r \rightarrow 0$ up (0.17 to 0.40), and the net rate *rises* from 31% to 50%, because shrinking σ_r removes persistent variance and lowers S/N from 1.30 to 0.90. Holding the ratio at 2 and scaling both components instead takes the rate from 31% to 8%. Second, **more transient inefficiency hurts**: σ_u enters the denominator, so a noisier transient block contaminates the firm means and makes the persistent split harder.

Note what this quantity does and does not explain. T enters it directly, so it covers the contamination channel by which longer panels help. It says nothing about N , which helps through a different route – sampling error in what is effectively a deconvolution from N firm-level draws – which is why the two dimensions have to be measured separately, as above.

One limit on all of that, stated because it is easy to over-read. Those elasticities were measured at $\sigma_r = 0.20$ against $\sigma_h = 0.40$, where persistent inefficiency is the larger of the two time-invariant components. On a design where the random effect dominates instead ($\sigma_r = 0.50$, $\sigma_h = 0.40$, $N = 100$, $T = 6$) the *level* ordering reverses: "f1ml" collapses on 33% of replications and "sml" on 20%. Which estimator handles the split better is not a fact about the estimators alone – it depends on how much persistent inefficiency there is to find.

psfm therefore reports the condition rather than preventing it, in `$sigh_at_bound` and `$sigr_at_bound` and in a warning naming the surviving scale. Bounding either parameter away from zero would corrupt precisely the samples where the boundary is the answer. What the warning asks of you is to read the two persistent scales together and to treat the split between them as unidentified in that sample – their *sum* remains informative even where the division does not.

Profiling a fitted panel likelihood. With `keep_objective = TRUE` the fitted object carries `$objective`, the same negative-summed-log-likelihood the optimizer minimised. Evaluating it away from the optimum is the only way to ask whether a parameter is weakly identified, as distinct from badly estimated:

```
fit <- psfm(y ~ x1 + x2, "GTRE", d, individual = "id",
  estimator = "sml", keep_objective = TRUE)
ih <- which(names(coef(fit)) == "sigh")
grid <- seq(0.05, 1, by = 0.05)
ll <- vapply(grid, function(v) {
  p <- fit$opt$par; p[ih] <- v; -fit$objective(p) }, numeric(1))
plot(grid, ll - max(ll), type = "l")
```

A flat curve means no estimator could recover the parameter from these data; a sharp one means the difficulty lies elsewhere. For GTRE's sigh the curve spans over 200 log units at $N = 200$ and sharpens with N , which is what ruled out weak identification as the explanation for its convergence behaviour.

The Schmidt–Sickles family. "SSFE", "SSRE" and "SSCRE" are the within, GLS and correlated-random-effects members of one family: all three read inefficiency off a firm effect as distance from the best firm, and none assumes a distribution for u . The choice among them is the standard panel trade-off, with a frontier-specific twist. Random effects is the more efficient estimator and is the only one of the three that identifies coefficients on *time-invariant* regressors, which the within transformation sweeps away – but it requires the effects to be uncorrelated with the regressors, and in a frontier model that assumption says a firm's inefficiency is unrelated to its input choices, which is close to saying managers do not respond to their own productivity.

"SSCRE" is the answer to that. Adding the within-firm means of the time-varying regressors (Mundlak 1978) models the correlation instead of assuming it away; the slopes on the original regressors then equal the within estimates *exactly* – not approximately – while time-invariant regressors remain identified. The coefficients on the added means are reported with a .mean_ prefix, and testing them jointly against zero is the Mundlak form of the Hausman test: if they are indistinguishable from zero, "SSRE" was safe. A regressor already constant within firm has a group mean identical to itself, so it contributes no Mundlak term; those are exactly the regressors random effects exists to identify, and they are left alone.

Kneip, Sickles and Song (2012). "KSS" writes the firm effect as $\alpha_{it} = \sum_{r=1}^L \theta_{ir} g_r(t)$, where both the basis functions g_r and the dimension L are estimated from the data rather than assumed. That makes explicit what the other time-varying estimators here are special cases of:

model	L	basis
"SSFE"	1	fixed, constant in t
"LS"	1	free
"CSS"	3	fixed to $\{1, t, t^2\}$
"KSS"	estimated	estimated

Estimation follows the three steps of the original: cross-sectional centering by period (which imposes $\sum_i u_i(t) = 0$ and removes any common function of time), then smoothing of each firm's residual trajectory followed by an eigendecomposition of their empirical covariance to give $\hat{g}_r$, then loadings by least squares and the effects rebuilt from them – iterated with β until the residual sum of squares settles. Smoothing before the eigendecomposition is what makes the estimated basis a set of smooth *functions* rather than T arbitrary period effects, and is the whole difference between this and an unrestricted factor model.

Two tuning choices are therefore unavoidable, and both are exposed. kss_smooth defaults to generalized cross-validation. kss_L defaults to the Bai and Ng (2002) IC_{p2} criterion – **not** the threshold rule of the original paper, which is stated here because it matters: IC_{p2} is the standard criterion for this problem and the literature that motivates the estimator treats Bai and Ng (2002) and Bai (2009) as the comparable methods, but a different rule can select a different L . Set kss_L explicitly to take the choice out of the criterion's hands, and inspect \$kss\$eigenvalues to see whether the choice was close.

Mundlak adjustment terms (mundlak). The true random effects model assumes the firm effect is independent of the regressors. In production data it usually is not – better managers choose different input levels – and the effect is then split between the slopes and inefficiency, biasing the technology

parameters and understating inefficiency. Mundlak's (1978) device models the correlation instead of assuming it away, $\alpha_i = \pi' \bar{x}_i + \delta_i$. Substituting into the frontier leaves the *same likelihood with a wider design matrix*, which is why this is an argument rather than a new model_name.

Karagiannis and Kellermann (2019) widen it to $\alpha_i = \pi' \bar{x}_i^* + \gamma' z_i + \delta_i$, adding environmental variables – conditions outside the producer's control, such as agro-ecological conditions or the geography an airline serves. Those enter the formula as ordinary time-invariant regressors; mundlak supplies the piece a user cannot easily build by hand, the within-firm means computed against the same panel index the fit uses. Naming a *subset* of the regressors gives their Hausman-Taylor variant, in which only some regressors are taken to correlate with the effect.

Measured over 40 replications with $N = 80$, $T = 6$, on a design where the firm effect is correlated with $\bar{x}$ by construction, and on a control where it is not:

design	estimator	bias	RMSE
effect correlated	plain	+0.0479	0.0570
effect correlated	mundlak	-0.0061	0.0310
no correlation	plain	-0.0033	0.0253
no correlation	mundlak	-0.0068	0.0311

So it removes the bias where there is one and nearly halves RMSE – and where there is no correlation it costs about 23% in RMSE for nothing. It is not free, and it is not a default.

A variable that is already constant within firm gets no mean column: its own mean is itself, so the two would be exactly collinear. Every mundlak fit also emits a collinearity warning from the *starting-value* regression, which is expected and harmless – in a between-individual regression x and $\bar{x}$ are the same column by construction. The starting values fall back to pooled OLS and the likelihood is estimated with the full requested formula.

The criterion needs a long enough panel, and says so. IC_{p2} works because the residual variance flattens out once the real factors are in and the penalty then dominates. On a short panel that never happens: if L is allowed up to $T - 1$ the factors can span nearly the whole time dimension, the variance collapses toward zero, and no penalty competes. Measured at $n = 100$ over five seeds and two true ranks, an uncapped criterion returned its maximum on *all ten* designs at $T = 6$ and again at $T = 8$. The automatic search is therefore capped at $\lfloor T/2 \rfloor$, which recovers the true rank exactly from $T = 8$ upward; at $T = 6$ it is right for a one-factor structure and occasionally one too high for a two-factor one. Whenever the automatic choice lands *on* that cap psfm warns, because a selection at the boundary is not a free choice – read it as "at least this many", inspect the eigenvalues, and prefer a longer panel to a larger `kss_L_max`.

One consequence of the period centering is worth knowing before comparing models: "KSS"'s `alpha_hat` carries no period mean, while "CSS"'s and "LS"'s do. The two therefore differ by a per-period constant by construction and should not be correlated against each other directly. `u_hat` is a within-period contrast and is invariant to that shift, so it is the quantity to compare across these models – and is what the estimators are for.

Time-varying fixed effects without a distribution for u. "CSS" and "LS" sit between "SSFE", which holds inefficiency fixed over time, and the maximum-likelihood panel models, which let it move but only by assuming a distribution for u and a decay path for it. Both instead let the firm effect itself vary with time and read inefficiency off it, so neither assumes any distribution and neither returns a log-likelihood.

They differ in what they are willing to spend on flexibility. Cornwell, Schmidt and Sickles (1990) give every firm its own quadratic in time, $\alpha_{it} = \theta_{i1} + \theta_{i2}t + \theta_{i3}t^2$: $3N$ parameters, firms may

overtake one another, and $T_i \geq 4$ is required before a firm contributes to β at all (a firm observed three times is fitted exactly by its own quadratic and its residuals are identically zero – sfa warns rather than letting such firms silently dilute the estimate). Lee and Schmidt (1993) impose one common temporal pattern scaled per firm, $\alpha_{it} = \delta_t \alpha_i$: $N + T - 1$ parameters, and the ranking of firms cannot change, only the spread. Cross-over versus no cross-over is the substantive choice between them, not a technical one.

"CSS" is fitted by residualizing y and X on each firm's own $(1, t, t^2)$ rather than by building $3N$ dummy columns; a regressor spanned by those quadratics – a pure time trend, for instance – is a rank failure and is refused rather than reported as a small coefficient. "LS" is a rank-one factor model, so it is fitted by alternating least squares, which handles an unbalanced panel with no special-casing; $\delta_1 = 1$ fixes the scale that $\delta_t \alpha_i$ leaves free, with a fallback to a norm constraint when the first period's effect is near zero.

For both, inefficiency is identified only relative to the best firm *in each period* – $u_{it} = \max_j \alpha_{jt} - \alpha_{it}$ – which is what makes them time-varying at all; comparing against a single overall best would collapse them back to a time-invariant ranking. Neither absorbs an intercept separately: it goes into the firm effect, as in "SSFE", so `coef()` returns slopes only.

For SSFE model:

<code>alpha_hat</code>	The estimated firm-specific fixed effects (named by individual).
<code>u_hat</code>	Estimated technical inefficiency for each firm, as its distance from the best-performing firm's fixed effect.
<code>exp_u_hat</code>	Predicted technical efficiency, <code>exp(-u_hat)</code> .

For the GTRE model:

<code>sigh_at_bound, sigr_at_bound</code>	TRUE when that persistent scale has collapsed to the zero boundary, judged relative to sigma. Frequently the CORRECT maximum likelihood estimate rather than a failure – see 'Details' – so it is reported rather than suppressed, and psfm warns naming the other persistent scale, since the two must be read together.
---	---

For CSS and LS models:

<code>alpha_hat</code>	The fitted firm effect for every observation, α_{it} – time-varying, unlike SSFE's one value per firm.
<code>u_hat, exp_u_hat</code>	Inefficiency and efficiency per observation, measured against the best firm in the same period.
<code>frontier_alpha</code>	The best firm effect in each period, i.e. the benchmark <code>u_hat</code> is measured from.
<code>sigma_v, residuals, df.residual</code>	The residual standard deviation, the residuals net of the firm effect, and the residual degrees of freedom ($NT - K - 3N$ for CSS, $NT - K - (N + T - 1)$ for LS).
<code>theta</code>	CSS only. An $N \times 3$ matrix of the firm-specific quadratic coefficients.
<code>delta, alpha_i, ls_iterations, ls_converged</code>	LS only. The common temporal pattern (with <code>delta[1] == 1</code>), the firm-specific scales, and the state of the alternating least-squares iteration.

For SSRE and SSCRE models:

<code>alpha_hat</code>	The BLUPs of the firm effects, one per firm. Under SSCRE these are net of the part the Mundlak means explain.
<code>theta, ercomp</code>	The GLS transformation parameter and the full variance-components decomposition from ercomp .
<code>mundlak_terms</code>	SSCRE only. The names of the added group-mean regressors.
<code>plm_fit</code>	The underlying plm object, for users who want its own methods.

For the KSS model:

<code>kss</code>	A list: the selected dimension L , the smoothing penalty κ , the estimated basis (a $T \times L$ matrix of orthonormal columns), the $n \times L$ matrix of loadings, the full eigenvalues sequence the dimension was chosen from, and the iteration state.
------------------	--

For PL80 and BC92 models:

<code>exp_u_hat</code>	Predicted technical efficiency for each observation, $E[\exp(-B_{it} * u_i) \epsilon_{it}]$ (generalizes Battese and Coelli's (1988, JoE) point estimator to the panel case, pooling all of a firm's observations to predict u_i and scaling by B_{it} per period).
------------------------	---

Note

Standard errors require `optHessian` set to TRUE

Note

The GTRE_SEQ1 and GTRE_SEQ2 models use sequential estimation methods and do not return optimization objects or starting values. The SSFE, CSS, LS, SSRE, SSCRE and KSS models are likewise not maximum-likelihood based and return no optimization object, starting values, or log-likelihood. All panel models require the `individual` argument to identify panel units.

Author(s)

David H. Bernstein

References

Fillipini and Greene (2016, JPA); Colombi, Kumbhakar, Martini and Vittadini (2014, JPA) on the closed-skew-normal four-component likelihood, with Colombi (2010) and Colombi, Martini and Vittadini (2011) on the two-step estimator used to seed it; Greene (2005, JoE and JPA); Wang and Ho (2010, JoE); Chen, Schmidt and Wang (2014, JoE); Kumbhakar (1990, JoE); Pitt and Lee (1980, JoE); Battese and Coelli (1992, JPA); Battese and Coelli (1988, JoE); Jondrow, Lovell, Materov and Schmidt (1982, JoE); Schmidt and Sickles (1984, JBES)

See Also

[sfm](#) for cross-sectional models, [psfm_bootstrap](#) for bootstrap standard errors, [data_gen_p](#) for simulating panel data with known true parameters, and [sfareg-methods](#) for extracting results.

Examples

```
library(sfa)

## A deliberately small panel with few simulation draws, so the examples
## build quickly. Do not read the estimates as a serious fit. TRE_Z becomes
## unstable below about 60 firms, so this is close to the floor.
data_trial <- data_gen_p(t=6, N=60, rand = 100,
                        sig_u = 1, sig_v = 0.3,
                        sig_r = .2, sig_h = .4,
                        cons = 0.5, beta1 = 0.5,
                        beta2 = 0.5)

## rand.psoptim is not optional with PSopt = TRUE: the particle swarm draws
## from the session RNG, so results otherwise change between builds.
max_tre_z <- psfm(formula = y_tre_z ~ x1 + x2 | z_gtre,
                  model_name = "TRE_Z",
                  data = data_trial,
                  individual = "name",
                  halton_num = 50,
                  rand.psoptim = 7L,
                  PSopt = TRUE)

## Time-varying fixed effects, no distribution assumed for u. CSS gives each
## firm its own quadratic in time; LS gives all firms one common pattern.
css_fit <- psfm(formula = y_tre ~ x1 + x2, model_name = "CSS",
                data = data_trial, individual = "name", time = "year")
ls_fit <- psfm(formula = y_tre ~ x1 + x2, model_name = "LS",
               data = data_trial, individual = "name", time = "year")
ls_fit$delta # the common temporal pattern, normalized to delta[1] = 1

## The Schmidt-Sickles family: within, GLS, and correlated random effects.
## SSCRE's slopes equal SSFE's within slopes exactly (Mundlak 1978); the
## .mean_ coefficients are the Mundlak form of the Hausman test.
ssre_fit <- psfm(formula = y_tre ~ x1 + x2, model_name = "SSRE",
                 data = data_trial, individual = "name", time = "year")
sscre_fit <- psfm(formula = y_tre ~ x1 + x2, model_name = "SSCRE",
                  data = data_trial, individual = "name", time = "year")

## Kneip, Sickles and Song (2012): the temporal basis and its dimension are
## both estimated. Inspect what the criterion chose before trusting it.
kss_fit <- psfm(formula = y_tre ~ x1 + x2, model_name = "KSS",
                data = data_trial, individual = "name", time = "year")
kss_fit$kss$L # selected dimension
kss_fit$kss$eigenvalues # was the choice close?
```

Description

Performs a parametric bootstrap for panel stochastic frontier models fit with `psfm`. Supports `model_name` `%in% c("GTRE_Z", "TRE_Z", "GTRE", "GTRE_FML", "TRE", "TFE", "FD")` – every `psfm()` model except the moment-based/LSDV estimators ("GTRE_SEQ1", "GTRE_SEQ2", "SSFE", which are not maximum-likelihood and have no `$opt` to bootstrap around in the same sense) and the `sfa`-wrapped models ("PL80", "BC92", whose `$out/efficiency-score` fields don't follow this package's own conventions). Each supported model's frontier/variance-component parameter blocks are read directly from `psfm_object$out` and `psfm_object$formula`; for each of `BOOT` replications, a new response is simulated from the fitted parameters using a data-generating process appropriate to that specific model (see Details), the model is re-estimated in parallel, and bootstrap standard errors / t-values are returned for every parameter in `psfm_object$out`.

Usage

```
psfm_bootstrap(psfm_object,
               numCores,
               BOOT,
               individual,
               h_type      = c("auto", "none", "scalar", "parametric"),
               maxit.psoptim = 1000,
               seed_offset  = 0,
               write_back   = TRUE,
               pkgs         = c("sfa", "Formula", "pbapply", "truncnorm"),
               inefdec,
               rand.gtre    = NULL,
               rand.psoptim = NULL,
               maxit.bobyqa = 1,
               maxit.optim  = 1)
```

Arguments

<code>psfm_object</code>	A fitted model object of class "sfareg" returned by <code>psfm</code> , with <code>model_name</code> <code>%in% c("GTRE_Z", "TRE_Z", "GTRE", "GTRE_FML", "TRE", "TFE", "FD")</code> . Note that <code>psfm(model_name = "GTRE")</code> returns <code>model_name = "GTRE_FML"</code> under its default estimator = "fml", which is why that value is supported here rather than only "GTRE". Must contain <code>\$out</code> , <code>\$data</code> , <code>\$formula</code> , and <code>\$model_name</code> ; <code>\$H</code> is additionally required for "GTRE"/"GTRE_FML"/"GTRE_Z" (the only models that return it) and <code>\$r_hat_m</code> is additionally required for "TFE".
<code>numCores</code>	Integer. Number of cores for the parallel cluster created internally via <code>makeCluster</code> .
<code>BOOT</code>	Integer. Number of bootstrap replications (e.g. 199, 399, 999).
<code>individual</code>	Character. Name of the individual/panel-ID column in <code>psfm_object\$data</code> (e.g. "firm.id").
<code>h_type</code>	One of "auto", "none", "scalar", or "parametric". Only meaningful for <code>model_name</code> <code>%in% c("GTRE_Z", "TRE_Z", "GTRE", "TRE")</code> (ignored for "TFE"/"FD", which have no random persistent-inefficiency h term); controls how the individual-specific, time-invariant inefficiency term h is simulated: "none" The model has no h term at all (TRE, TRE_Z).

	"scalar" $h_i \sim \text{abs}(N(0, \text{sigma}_h))$, a single scalar standard deviation – used automatically for bare "GTRE" (which has no h-covariates at all, only a single fitted sigh). "parametric" $h_i \sim \text{abs}(N(0, \text{exp}(\text{data}_h \% \% h_vec)))$, using the third right-hand-side part of the model formula as the h-covariates – used automatically for "GTRE_Z". "auto" (Default) Resolves per model_name: "parametric" for "GTRE_Z", "scalar" for bare "GTRE" and "GTRE_FML", "none" for "TRE_Z"/"TRE".
maxit.psoptim	Integer. Passed through to psfm() as maxit.psoptim on each bootstrap re-estimation. Default 1000.
seed_offset	Integer added to the bootstrap index b when calling set.seed(), allowing reproducible-but-distinct seeds across multiple bootstrap runs. Default 0.
write_back	Logical. If TRUE (default), the returned list includes model, a copy of psfm_object with the bootstrap standard errors and t-values written into \$out[, 2] and \$out[, 3]. If FALSE, only the raw bootstrap results are returned.
pkgs	Character vector of package names to load on each cluster worker via clusterCall. Default c("sfa", "Formula", "pbapply", "truncnorm") – truncnorm is only actually used when bootstrapping an "FD" fit, but is included in the default so a single default works for every supported model.
inefdec	Logical, with no default; must be supplied. Passed through to psfm() on each bootstrap re-estimation. Should match the inefdec setting used to originally fit psfm_object.
rand.gtre	Passed through to psfm() as rand.gtre on each bootstrap re-estimation. Default NULL.
rand.psoptim	Passed through to psfm() as rand.psoptim on each bootstrap re-estimation. Default NULL.
maxit.bobyqa	Integer. Passed through to psfm() as maxit.bobyqa on each bootstrap re-estimation. Default 1.
maxit.optim	Integer. Passed through to psfm() as maxit.optim on each bootstrap re-estimation. Default 1.

Details

Supported models fall into three families, each with its own data-generating process and \$out row layout (see R/psfm_bootstrap.R's header comment for the full derivation, including how each was cross-checked against that model's own likelihood in psfm.R):

"randeff" **family** (GTRE_Z, TRE_Z, GTRE, TRE) $y^* = X \% \% \text{beta_hat} + v + u + r + h$ (sign per inefdec), with $v \sim N(0, \text{sigma}_v^2)$, u half-normal (covariate-driven sigma for GTRE_Z/TRE_Z, a single homoskedastic scalar for bare GTRE/TRE, which have no z pipe at all), $r \sim N(0, \text{sigma}_r^2)$ (one draw per individual, repeated over time), and h per h_type (GTRE/GTRE_Z only). \$out row layout: sigv, sigr, [x-block], [z-block], [h-block] for GTRE_Z/TRE_Z; lambda, sigma, sigr, [sigh], [x-block] for bare GTRE/TRE (note the different reparameterization – see psfm's own model-family sigma-link inconsistency note).

"tfe" **family** (TFE) Fixed-effects model: the individual effects are held **FIXED** at `psfm_object$r_hat_m`'s original point estimates across every replication (not redrawn – redrawing them would contradict treating them as fixed rather than random), and only $v \sim N(0, \sigma_v^2)$ and $u \sim |N(0, \sigma_u^2)|$ (both derived from the fitted λ/σ , or γ/σ if the original fit used `psfm(..., gamma = TRUE)`) are redrawn. \$out row layout: λ (or γ), σ , [x-block] (no intercept row – the within-transformation removes it).

"fd" **family** (FD) Wang and Ho (2010) first-difference estimator: a single time-invariant $u_i \sim \text{TruncatedNormal}(\text{mean} = \mu, \text{sd} = \sqrt{\text{sig_u2}}, \text{lower} = 0)$ draw per individual (via `rtruncnorm`), scaled by a deterministic time-varying factor $h_{it} = \exp(z_{it} \% \delta)$, plus $v_{it} \sim N(0, \text{sig_v2})$. \$out row layout: sig_u2 , sig_v2 , μ , [x-block], [z-block] (no intercept row in either the x- or z-block).

`psfm_bootstrap` validates that `nrow(psfm_object$out)` matches the expected layout for the resolved model/`h_type` before running any bootstrap replications, and stops with an informative error if it does not. For `GTRE/GTRE_Z` it similarly validates up front that `length(psfm_object$H)` equals the number of unique individuals implied by the individual column (`$H` is one value per individual, not one per observation) – `TRE/TRE_Z/TFE/FD` do not return `$H` at all, so `boot_eff_h` is simply `NULL` for those four models (not an error).

The model is re-estimated on each simulated dataset via `sfa::psfm()`, executed in parallel across replications using `parLapply` (or `pblapply` for a live progress bar, when the `pbapply` package is installed). Whether the `psoptim` (particle-swarm) optimizer stage is used on each refit is model-dependent: it is enabled for the "randeff" family (matching this function's original behavior, tested clean), but disabled for "TFE" – testing found `psoptim`'s wide exploration made TFE repeatedly collapse onto a degenerate variance-boundary solution (reproduced even refitting the SAME real data across different `psoptim` seeds), a problem that disappeared entirely once that stage was skipped (matching `psfm()`'s own default of `PSopt = FALSE`). As a defense-in-depth safety net across every family, any replication whose refit still lands on a degenerate variance-boundary mode (a scale parameter pinned near its numerical floor – the same well-understood boundary-mode phenomenon documented for `ttsfm()`'s `TTNE/TTN` models) is treated as a per-replication **FAILURE** rather than silently corrupting the bootstrap standard error with a meaningless draw.

Re-estimation failures (optimizer errors, or the degenerate-mode check just described) are caught per replication, recorded with their message, and set to `NA` in the output matrices rather than stopping the bootstrap; a single **warning** at the end lists every failed replication index together with its message. Bootstrap standard errors are computed with `na.rm = TRUE`, so they remain well-defined even when some replications failed.

Requires the **parallel** and **Formula** packages (and **truncnorm** when bootstrapping an "FD" fit), and requires **sfa** (and any other packages named in `pkgs`) to be installed and loadable on each cluster worker.

Value

A list with components:

<code>boot_par</code>	A <code>B00T</code> x <code>(nrow(psfm_object\$out) + 2)</code> matrix of bootstrapped parameter draws, with the final two columns "loglik" and "hours". Rows corresponding to failed replications are <code>NA</code> .
<code>boot_eff</code>	A <code>B00T</code> x <code>nrow(psfm_object\$data)</code> matrix of bootstrapped per-observation efficiency-score draws (<code>\$U</code> for the "randeff" family, <code>\$exp_u_hat</code> for "TFE"/"FD")

	– the field is named differently across these models in psfm.R, but always collected into this same boot_eff matrix here). Rows corresponding to failed replications are NA.
boot_eff_h	Only non-NULL for model_name %in% c("GTRE", "GTRE_FML", "GTRE_Z") (the only models that return \$H at all); NULL otherwise. A BOOT x length(psfm_object\$H) matrix of bootstrapped time-invariant, individual-specific inefficiency-score (\$H) draws – one column per unique individual (not one column per observation; \$H does not vary over time within an individual). Columns are named by the individual ID values. Rows corresponding to failed replications are NA.
se	Named numeric vector of bootstrap standard errors for every parameter in psfm_object\$out.
tval	Named numeric vector of bootstrap t-values (estimate / boot SE) for every parameter in psfm_object\$out.
failures	Integer vector of replication indices b for which re-estimation failed, or NULL if none failed.
model	Only present when write_back = TRUE. A copy of psfm_object with \$out[,2] and \$out[,3] replaced by the bootstrap standard errors and t-values.

Author(s)

David H. Bernstein and Christopher F. Parmeter

See Also

[psfm](#), [parLapply](#), [pblapply](#), [Formula](#)

Examples

```
## Not run:
## Not run: a meaningful bootstrap needs BOOT in the hundreds, each
## replication refitting the model on a PSOCK cluster, so this takes
## minutes rather than seconds.

data_trial <- data_gen_p(t = 5, N = 50, rand = 100,
                        sig_u = 1, sig_v = 0.3,
                        sig_r = 0.2, sig_h = 0.4,
                        cons = 0.5, beta1 = 0.5, beta2 = 0.5)

GTRE_ZZ <- psfm(y_gtre_zz ~ x1 + x2 | z_gtre | zp_gtre,
               model_name = "GTRE_Z",
               data       = data_trial,
               individual = "name")

boot_results <- psfm_bootstrap(
  psfm_object = GTRE_ZZ,
  numCores    = 2,
  BOOT        = 199,
  individual  = "name",
  h_type      = "auto",
  inefdec     = FALSE
)
```

```
## Bootstrap standard errors / t-values for every model parameter
boot_results$se
boot_results$tval

## Bootstrap draws of time-varying (U) and time-invariant (H)
## efficiency/inefficiency scores, one row per replication
boot_results$boot_eff
boot_results$boot_eff_h

## Fitted object with bootstrap SEs/t-values written into $out
boot_results$model$out

## End(Not run)
```

selsfm	<i>Stochastic Frontier Models with Sample Selection or Technology Choice</i>
--------	--

Description

Two models, chosen by `model_name`.

"greene" (the default) fits Greene's (2010) sample-selection stochastic frontier: a normal-half normal frontier observed only for the subset of units that select into the sample, where the unobservables driving selection are correlated with the *noise* in the frontier. Estimation follows the two-step scheme of the paper – a probit for the selection equation on the full sample, then maximum *simulated* likelihood for the frontier on the selected subsample.

"kts" fits Kumbhakar, Tsionas and Sipilainen (2009), which is a *different* model rather than a variant. Both technologies are observed, each with its own frontier and its own two scales, and the technology choice depends on **inefficiency itself**: u lowers output through the frontier and shifts the adoption decision through the choice equation. Fitted by single-step maximum likelihood over the whole sample.

Usage

```
selsfm(selection, frontier, data, model_name = c("greene", "kts"),
        n_nodes = 64, Nsim = "auto",
        sim_type = c("halton", "sobol", "torus", "uniform"),
        antithetics = FALSE, seed = NULL, inefdec = TRUE,
        maxit.bobyqa = 10000, maxit.psoptim = 1000, maxit.optim = 1000,
        start_val = FALSE, PSopt = FALSE, optHessian = TRUE,
        Method = "L-BFGS-B", verbose = FALSE, rand.psoptim = NULL)
```

Arguments

selection	A two-sided formula for the selection equation, $d \sim z_1 + z_2$. The response must be binary (0/1, logical, or a two-level factor) and is modelled by probit over <i>all</i> rows of data.
-----------	--

frontier	A two-sided formula for the frontier, $y \sim x_1 + x_2$, fitted on the selected rows only. The dependent variable need not be observed where $d == 0$; those rows may be NA.
model_name	"greene" (default) or "kts"; see 'Description' and 'The KTS model' below.
n_nodes	Gauss–Legendre nodes for the integral over u , used by "kts" only. The half-normal integral is mapped to $[0, 1]$ by its own CDF, so no truncation point is needed.
data	A data.frame containing both equations' variables, with the unselected rows present. Unlike the rest of the package neither formula takes a segment – the two equations are separate arguments – and a pipe is an error rather than being silently ignored.
Nsim	Number of simulation draws per selected observation. "auto" (the default) uses $\max(200, \text{ceiling}(3 \cdot \sqrt{n_{\text{selected}}}))$. Simulated ML is consistent only if the draw count grows with the sample size; a numeric value below the automatic floor warns.
sim_type	Low-discrepancy sequence for the draws, passed to the same generator the other simulated-ML models use.
antithetics	Mirror the draws to halve the number of independent points.
seed	Optional seed for the draws. The RNG state is saved and restored, so a fit does not disturb the caller's stream.
inefdec	TRUE (default) for a production frontier, where inefficiency is subtracted; FALSE for a cost frontier.
maxit.bobyqa, maxit.psoptim, maxit.optim	Iteration caps for the three optimizer stages.
start_val	Name the starting-value vector in the returned object.
PSopt	Run the particle-swarm stage between BOBYQA and optim.
optHessian	Compute the Hessian, and with it the standard errors.
Method	Method passed to optim for the final stage.
verbose	Report optimizer progress.
rand.psoptim	Optional seed for the particle-swarm stage.

Details

The model is Greene's Equation (10):

$$\begin{aligned}
 d_i &= 1[\alpha' z_i + w_i > 0], \quad w_i \sim N(0, 1) \\
 y_i &= \beta' x_i + v_i - u_i, \quad \text{observed only when } d_i = 1 \\
 u_i &= \sigma_U |U_i|, \quad U_i \sim N(0, 1), \quad v_i = \sigma_V V_i, \quad V_i \sim N(0, 1) \\
 (w_i, v_i) &\sim N_2[(0, 0), (1, \rho\sigma_V; \rho\sigma_V, \sigma_V^2)]
 \end{aligned}$$

Selection operates through the *noise* v , not through the inefficiency u . That is the substantive assumption separating this model from Kumbhakar, Tsionas and Sipilainen (2009), where the selection disturbance is correlated with u instead; the two answer different questions and are not nested.

Because u cannot be integrated out in closed form, the log-likelihood is simulated (Greene's Equation (15)): for each selected observation R draws of $|U_{ir}|$ are taken and

$$\log L = \sum_{d_i=1} \log \frac{1}{R} \sum_{r=1}^R \phi\left(\frac{v_{ir}}{\sigma_V}\right) \frac{1}{\sigma_V} \Phi\left(\frac{\rho v_{ir}/\sigma_V + \alpha' z_i}{\sqrt{1-\rho^2}}\right)$$

with $v_{ir} = y_i - \beta' x_i + \sigma_U |U_{ir}|$. Everything is evaluated in logs and accumulated by log-sum-exp, since the summand underflows in the tails well before the total does.

A note on the published formula. Equations (11) and (13)–(15) of Greene (2010) write σ_ε in the denominator of the Φ argument. It must be σ_V : the term being standardized is $v_{ir} = y_i - \beta' x_i + \sigma_U |U_{ir}|$, which is the noise, and the conditional distribution of w given v for a bivariate normal with $\text{Var}(w) = 1$ and correlation ρ is $N(\rho v/\sigma_V, 1 - \rho^2)$. This implementation uses σ_V .

Standard errors are conditional on the first stage. Following the paper's two-step scheme, α is taken as given in the frontier likelihood. The reported standard errors come from the second-stage Hessian and therefore do *not* include the sampling variability of $\hat{\alpha}$; Greene applies the Murphy–Topel correction at this point, which is not implemented here. Treat them as a lower bound, and bootstrap over both stages when the precision matters.

Value

An object of class "sfareg". As elsewhere in the package out is a $p \times 3$ matrix, one row per parameter, with columns par, st_err and t-val; the parameters are ordered sigma_u, sigma_v, rho, then the frontier coefficients. Alongside the usual components:

alpha	First-stage probit coefficients.
probit	The fitted first-stage object.
jlms	$E[u_i \mid \varepsilon_i]$ by the simulated Bayes rule of Greene's Equation (21), computed over the same draws the likelihood used.
efficiency	$\exp(-jlms)$.
selection_formula	The selection equation.
n_selected, n_total	Selected and total observation counts.
Nsim, sim_type	The draw count and sequence actually used.

The KTS model

For technology $I_i \in \{0, 1\}$,

$$y_i \mid I_i, u_i \sim N(x_i' \beta_{I_i} - u_i, \sigma_{v, I_i}^2), \quad P(I_i = 1 \mid u_i) = \Phi(z_i' \gamma + \delta u_i), \quad u_i \mid I_i \sim N^+(0, \sigma_{u, I_i}^2).$$

δ is the parameter the model exists for: it is the *dual role* of inefficiency, and $\delta > 0$ means the less efficient are more likely to choose technology 1. Neither two-step order works – the choice equation cannot be a probit because u is unobserved, and fitting the frontiers first ignores the endogeneity of the choice – which is why the paper estimates in one step.

Because u 's distribution depends on I while I 's probability depends on u , the marginal density of u is a *mixture* of the two regimes' half-normals, with weight ϕ_i solving the paper's equation (9). The observation's density is then the same mixture of two one-dimensional integrals, its equation (14).

Recovery. On data simulated from this exact model with $\delta = 0.9$, the frontier coefficients, the four scales and δ all recover, and improve with the sample:

parameter	n = 1500	n = 6000	truth
δ	1.361	0.922	0.900
$\sigma_{u,0}$	0.837	0.843	0.800
$\sigma_{u,1}$	0.632	0.640	0.600
choice intercept	-0.511	-0.239	-0.200

δ is the parameter that needs the most data, as one would expect of a coefficient identified only through the coupling between the two equations, but it is properly identified: holding the rest at the truth, the mean log-likelihood falls monotonically away from $\delta = 0.9$ in both directions, by about 0.069 per observation across $[0.1, 1.8]$.

References

- Kumbhakar, S.C., Tsionas, E.G. and Sipilainen, T. (2009). Joint estimation of technology choice and technical efficiency: an application to organic and conventional dairy farming. *Journal of Productivity Analysis*, 31(3), 151–161.
- Greene, W. (2010). A stochastic frontier model with correction for sample selection. *Journal of Productivity Analysis*, 34(1), 15–24.
- Heckman, J. (1979). Sample selection bias as a specification error. *Econometrica*, 47(1), 153–161.

See Also

[sfm](#) for the frontier without a selection correction.

Examples

```
set.seed(1)
n <- 800
z1 <- rnorm(n); x1 <- rnorm(n)
w <- rnorm(n)
d <- as.numeric(0.2 + 0.8 * z1 + w > 0)
v <- 0.5 * (0.6 * w + sqrt(1 - 0.6^2) * rnorm(n))
y <- 0.5 + 0.8 * x1 + v - abs(rnorm(n))
y[d == 0] <- NA
dat <- data.frame(y = y, x1 = x1, z1 = z1, d = d)

fit <- selsfm(selection = d ~ z1, frontier = y ~ x1, data = dat, Nsim = 100)
fit$out
```

Description

Standard extractor methods for objects of class "sfareg", as returned by `psfm`, `sfm`, `zsfm`, and `ttsfm`. These let "sfareg" fits be used with the usual R modeling idioms (e.g. `coef(fit)`, `logLik(fit)`, `AIC(fit)`, `BIC(fit)`) instead of reaching into the returned list by hand.

`AIC()` and `BIC()` are not defined explicitly for "sfareg" objects; they fall through to their default methods in the **stats** package, which only require a working `logLik` method (and, for `BIC()`, a working `nobs` method) to work correctly.

Usage

```
## S3 method for class 'sfareg'
coef(object, ...)
## S3 method for class 'sfareg'
vcov(object, type = c("hessian", "bhhh"), ...)
## S3 method for class 'sfareg'
logLik(object, ...)
## S3 method for class 'sfareg'
nobs(object, ...)
```

Arguments

<code>type</code>	For <code>vcov</code> : "hessian" (the default) for the inverse Hessian, or "bhhh" for the inverse outer product of the per-observation scores. See 'Details'.
<code>object</code>	An object of class "sfareg".
<code>...</code>	Currently unused; included for S3 method consistency.

Details

BHHH standard errors. `vcov(fit, type = "bhhh")` returns the inverse of the outer product of the per-observation scores instead of the inverse Hessian. Its value is that it needs no Hessian at all: it is defined whenever the scores are, which is exactly the case where the default path fails. When the Hessian is singular `vcov` otherwise falls back to a *diagonal* approximation built from the reported standard errors, discarding every covariance; BHHH keeps them. It requires `keep_objective = TRUE` for the same reason `estfun` does. It is not the default, because it is only as good as the information-matrix equality and so disagrees with the Hessian under misspecification – which is useful when the two are compared on purpose and misleading when they are not.

Robust and clustered standard errors. `vcov` returns the inverse Hessian, which is valid only if the likelihood is correctly specified and the observations are independent – neither being safe to assume in applied frontier work. Registering `bread` and `estfun` makes the **sandwich** package operate on these fits, so `sandwich::sandwich()` gives Huber–White errors and `sandwich::vcovCL()` gives clustered ones, and either can be handed to `lmtest::coeftest()`:

```
fit <- sfm(y ~ x1 + x2, model_name = "NHN", data = d,
          keep_objective = TRUE)
lmtest::coeftest(fit, vcov. = sandwich::vcovCL(fit, cluster = d$firm))
```

Two requirements. The fit must have been made with `keep_objective = TRUE`, since the scores are differenced from the retained likelihood; and the robust divergence estimators are excluded, because they do not maximise a log-likelihood and so have no score. `sandwich::vcovHC()` does *not* work and is not expected to: its finite-sample corrections are built from hat values, which are not defined for a nonlinear likelihood whose parameter vector includes variance components.

The scores are taken by central differences with a per-parameter step, there being no analytic gradient for most of these models. Where one does exist (NHN) the two agree to about 3×10^{-7} , and they agree with `numDeriv` to the same order.

`coef.sfareg` returns the named vector of estimated parameters (`object$coefficients`).

`vcov.sfareg` returns the estimated variance-covariance matrix. If the fit was estimated with `optHessian = TRUE` and the Hessian is invertible, this is `solve(object$opt$hessian)`. Otherwise it falls back to a diagonal matrix built from the reported standard errors (`object$std.errors`), with a warning; if neither is available it returns a matrix of NAs with a warning.

`logLik.sfareg` returns the maximized log-likelihood as an object of class "logLik", with "df" and "nobs" attributes set so that `AIC()/BIC()` work automatically. Every estimator in this package minimizes the negative summed log-likelihood internally, so `logLik = -object$opt$value`. **Exception:** `psfm()`'s "GTRE_SEQ1" and "GTRE_SEQ2" models are estimated by a moment-based procedure and "SSFE" by fixed-effects (LSDV) regression, not by maximum likelihood; none of the three carries an `$opt` component, so calling `logLik()` (or, transitively, `AIC()/BIC()`) on those fits returns NA with a warning rather than erroring.

`nobs.sfareg` returns the number of observations used in estimation, recovered from `object$data` when the fitting function stored it, or by re-evaluating the data argument of the original call otherwise. Returns NA if neither is available.

Value

`coef.sfareg` returns a named numeric vector.

`vcov.sfareg` returns a numeric matrix.

`logLik.sfareg` returns an object of class "logLik".

`nobs.sfareg` returns a single integer (or NA).

See Also

[psfm](#), [sfm](#), [zsfm](#), [ttsfm](#), [print.sfareg](#), [summary.sfareg](#)

Description

Reports how the optimizer actually finished and how well the likelihood identifies the parameters: the convergence code and message, the eigenvalue spectrum and conditioning of the Hessian, the implied parameter correlations, and – where the objective was retained – the gradient at the reported optimum and a likelihood slice through each parameter.

Usage

```
sfa_diagnostics(object, ...)

## S3 method for class 'sfareg'
plot(x, which = 1:4, n_grid = 41, span = 0.25, ...)

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

Arguments

object, x	An object of class "sfareg" (for sfa_diagnostics and plot), or of class "sfadiag" (for print).
which	Which panels to draw: 1 the Hessian eigenvalue spectrum, 2 the parameter correlation matrix, 3 a likelihood slice per parameter, 4 the gradient. Panels that cannot be drawn are dropped silently; asking for <i>only</i> unavailable panels is an error rather than an empty plot.
n_grid	Number of points in each likelihood slice. Defaults to 41.
span	Half-width of each slice, as a fraction of $\max(\text{estimate} , 1)$. Defaults to 0.25.
...	Currently unused.

Details

Every stochastic frontier fit in this package runs a staged minimizer and stores `optim()`'s convergence code, message, evaluation counts and numerically differentiated Hessian. None of that was reported: a fit that stopped on the iteration cap printed exactly like a converged one. `sfa_diagnostics()` is the reporting side of that.

What the Hessian tells you. `optim()` returns the Hessian of the *negative* log-likelihood, so at a genuine interior optimum it is positive definite. A non-positive eigenvalue means the reported point is not a minimum in that direction. A very small one means the likelihood is nearly flat along it, which is what weak identification looks like numerically – `$hessian$flat_direction` names the parameters that load on that flattest axis, and they are usually the pair the data cannot separate.

Slices, not profiles. Panel 3 varies one parameter and holds the others **at their estimates**. A true profile would re-optimize the remaining parameters at every grid point, at the cost of one

optimization per point. The slice is the right object for reading curvature at the optimum and for the simplest check there is: the estimate should sit at the peak of its own slice. If it does not, the optimizer stopped early.

Retaining the objective. The gradient and the slices need the likelihood itself, which an ordinary fit does not keep. Pass `keep_objective = TRUE` to `sfm` to store it. This is off by default because a closure carries its enclosing environment, so a fit saved with one serializes the estimation data alongside the results – on a 200-observation example the fitted object grows from roughly 38 KB to 1.7 MB.

The convergence code alone is not diagnostic, and the flags do not treat it as though it were. Across NHN, NE and NTN at $n = 150, 500$ and 1500 , code 52 (“ABNORMAL_TERMINATION_IN_LNSRCH”) turns up routinely alongside a maximum relative gradient of about 10^{-6} and a positive definite Hessian: the staged minimizer had already found the optimum and the final L-BFGS-B stage simply could not take a step away from it. The same code on NTN at $n = 150$ came with a relative gradient of 5×10^7 and an indefinite Hessian, which is a genuine failure. What separates the two is the gradient and the Hessian, so the verdict is built from those and the code is reported as context:

benign a line-search code (51/52) together with a small relative gradient and a positive definite Hessian. Reported as noise.

unverified a line-search code with no retained objective, so there is no gradient to settle it. Reported as an open question rather than resolved in the fit’s favour – refit with `keep_objective = TRUE`.

failure anything else, including code 1 (the iteration limit), which is never excused: it means the search was still moving when it ran out of budget, whatever the curvature at the stopping point.

Flags are raised for a non-zero convergence code, a Hessian that is not positive definite or is worse than 10^8 conditioned, any parameter pair correlated beyond $|0.95|$, and a large relative gradient. They are phrased as statements about what the fit means, not as generic warnings.

Value

`sfa_diagnostics()` returns an object of class “sfadiag”: a list with `model_name`, `call`, `pnames`, `estimates`, `convergence` (code, meaning, message, counts, logLik), `hessian` (eigenvalues, condition, pos_def, flat_direction), `correlation` (matrix, worst_pair, worst_value); `convergence` additionally carries `benign_nonzero` and `unverified_nonzero`, `gradient` (gradient, max_abs, relative, max_rel; NULL without a retained objective), `flags`, and `has_objective`.

`plot.sfareg()` is called for its side effect and returns the “sfadiag” object invisibly. `print.sfadiag()` returns its argument invisibly.

Author(s)

David H. Bernstein

See Also

[sfm](#), [sfareg-methods](#), [summary.sfareg](#)

Examples

```
library(sfa)
d <- data_gen_cs(N = 300, rand = 3, sig_u = 1, sig_v = 0.3,
                 cons = 0.5, beta1 = 0.5, beta2 = 0.5, a = 5, mu = 0.5)

## Retaining the objective enables the gradient and the likelihood slices
f <- sfm(y_pcs ~ x1 + x2, model_name = "NHN", data = d, keep_objective = TRUE)
sfa_diagnostics(f)

## Four-panel diagnostic display
plot(f)

## Works without the objective too, on the Hessian alone
g <- sfm(y_pcs ~ x1 + x2, model_name = "NHN", data = d)
sfa_diagnostics(g)$hessian$condition
```

sfm

sfm

Description

Implementation of the cross-sectional stochastic frontier model across an array of distributional assumptions for both v and u (user specified). For panel models, see the `psfm()` call.

Usage

```
sfm(formula, model_name = c("NHN", "NHN_Z", "NE", "NE_Z", "NR",
                             "THT", "NTN", "NG", "NNAK", "NU", "NGE", "NLN", "NW", "tHN",
                             "TSL"), data, maxit.bobyqa = 10000, maxit.nlminb = 500,
    maxit.psoptim = 1000, maxit.optim = 1000, REPORT = 1, trace = 2,
    pgtol = 0, start_val = FALSE, PSopt = FALSE, use.nlminb = "auto",
    use.bobyqa = "auto", optHessian = TRUE, inefdec = TRUE, upper =
    NA, Method = "L-BFGS-B", robust = c("mle", "mlqe", "psi",
    "mdpd"), c_mlqe = 0.2, eta = 0.01, alpha = 0.2, verbose = FALSE,
    start_from = NULL, Nsim = "auto", z_link = c("sd", "var"), vhet =
    NULL, uheter = NULL, muhet = NULL, scaling = NULL, shapehet = NULL,
    weights = NULL, wscale = TRUE, sim_type = c("halton", "sobol",
    "torus", "uniform"), antithetics = FALSE, sim_burn = NULL,
    sim_scrambling = 0L, sim_prime = NULL, sim_seed = NULL,
    rand.psoptim = NULL, keep_objective = FALSE, estimator = c("mle",
    "cols", "mols"), cols_boot = 0, rand.cols = NULL)
```

Arguments

<code>formula</code>	a symbolic description for the model to be estimated
<code>model_name</code>	Which distributional pair to fit. The noise term v is normal except where stated, and the one-sided inefficiency term u follows:

	"NHN" half normal.
	"NHN_Z" half normal with heteroskedastic σ_u , parameterized by the determinants z given in the second part of the formula ($y \sim x \mid z$) as $\sigma_u = \exp(z'\delta)$.
	"NE" exponential.
	"NE_Z" exponential with heteroskedastic σ_u , parameterized as for "NHN_Z".
	"NTN" truncated normal.
	"NR" Rayleigh. Note this is a distinct family from "NHN", not a reparameterization of it: the Rayleigh density carries an extra factor of u , and its standardized skewness is fixed at -0.631 against the half normal's -0.996.
	"NU" uniform.
	"NG" gamma. See <code>ng_starts</code> under 'Value' – this likelihood determines $E[u] = \mu\sigma_u$ far better than either factor separately.
	"NNAK" Nakagami, with shape m . Nests "NR" at $m = 1$. The shape is weakly identified in practice, so σ_u and m are strongly correlated across samples.
	"NGE" generalized exponential.
	"NLN" lognormal. Estimated by simulated maximum likelihood; see <code>Nsim</code> and 'Details'.
	"NW" Weibull. Also simulated maximum likelihood.
	"tHN" half normal, with Student- t noise.
	"TSL" truncated skew-Laplace.
	"THT" half t , with Student- t noise.
	Matching ignores case, so "nhn", "NHN" and "Nhn" are the same choice.
<code>data</code>	A data frame containing the variables named in formula.
<code>maxit.bobyqa</code>	Maximum number of iterations for the bobyqa optimization routine
<code>maxit.psoptim</code>	Maximum number of iterations for the psoptim optimization routine
<code>maxit.optim</code>	Maximum number of iterations for the optim optimization routine
<code>REPORT</code>	reporting parameter
<code>trace</code>	Integer. Tracing level passed through to the optimizer; larger values print more.
<code>pgtol</code>	Numeric. Projected-gradient tolerance passed to <code>optim()</code> 's "L-BFGS-B" method. $\emptyset$ uses the optimizer's own default.
<code>start_val</code>	starting value (optional)
<code>PSopt</code>	use psoptim optimization routine (T or F)
<code>maxit.nlminb</code>	Maximum number of iterations for the nlminb optimization routine.
<code>use.nlminb</code>	"auto" (default), or a logical. Whether to run the <code>stats::nlminb</code> stage, which precedes bobyqa in the optimizer stack and accepts an analytic gradient where one is available. Under "auto" it runs only for the models where it has been verified to reach an equal-or-better optimum than the previous bobyqa-based path (NHN, NE, NTN, NU), where it is roughly 4-8 times faster at identical accuracy.
<code>use.bobyqa</code>	"auto" (default), or a logical. Whether to run the derivative-free bobyqa stage. Under "auto" it runs for every model <i>except</i> those handled by nlminb above, reproducing the previous behaviour exactly. It is kept on by default for the remaining models deliberately: for NR and NGE the nlminb path converges to a materially worse optimum, so accuracy, not speed, decides which stage runs.

optHessian	Logical. Should a numerically differentiated Hessian matrix be returned while using the optim routine? (for optim routine)
inefdec	Production or cost function
upper	Vector of upper values for the optim package.
Method	The method to be used for optim. See 'Details' within optim.
robust	One of "mle" (default), "mlqe", "psi", or "mdpd". Selects a robust divergence-based objective in place of ordinary maximum likelihood – see Details. Currently only implemented for model_name = "NHN"; sfm() stops with a clear error if a non-"mle" value is requested for any other model.
c_mlqe	Numeric, default 0.20. The MLqE power parameter (c in f^c ; the corresponding $q = 1-c$), used only when robust = "mlqe".
eta	Numeric, default 0.01. The Psi-likelihood power parameter, used only when robust = "psi".
alpha	Numeric, default 0.2. The MDPD (minimum density power divergence) power parameter, used only when robust = "mdpd".
verbose	Logical. Print optimization progress messages? Default is FALSE.
weights	Optional observation weights, one per row of the data actually fitted (that is, after missing-data handling; a length mismatch is an error rather than being recycled). They multiply the per-observation log-likelihood contributions. For frequency weights this is exact – a weight of 2 gives the same estimates as including the row twice, which the package tests – and for sampling weights it yields a pseudo-likelihood whose point estimates are consistent but whose Hessian-based standard errors are not. In that case use the sandwich methods this package registers, e.g. <code>sandwich::sandwich(fit)</code> . Cannot be combined with the robust divergence estimators, which reweight observations themselves.
wscale	Rescale weights to sum to the number of observations. This puts the log-likelihood on the same scale as an unweighted fit, so AIC/BIC stay comparable; it is a common factor on the objective and does not move the estimates. Set FALSE for genuine frequency weights, where the unscaled total is the quantity of interest.
shapehet	Experimental. Optionally, a one-sided formula parameterizing the <i>shape</i> of u rather than its scale, as $\mu_i = \mu \exp(z_i' \delta)$. Implemented for model_name = "NG" and "NNAK", the two families that have a shape parameter at all. On a design with a true shape effect it improves the log-likelihood by a very large margin (about 10,000 points at $n = 2000$), so it does capture the variation. But its parameters do not recover well in testing: on that design the shape coefficient came back at 0.19 against a true 0.4 while σ_V collapsed to its lower bound, which is the signature of a degenerate optimum rather than an estimate. Plain "NG" recovers cleanly on the same design without a shape effect, so this is specific to the extended model and not to "NG" itself. Treat the coefficients as exploratory, check σ_V for a boundary collapse before reading them, and prefer uh _t where scale heterogeneity is what you actually mean.
scaling	Optionally, a one-sided formula of covariates imposing the scaling property of Wang and Schmidt (2002) and Alvarez, Amsler, Orea and Schmidt (2006): $u_i = h(z_i, \delta) u_i^*$ with $h = \exp(z_i' \delta)$, so the covariates scale a common draw

u^* rather than entering its variance. Both σ_U and the pre-truncation mean move by the *same* factor, which holds the shape of the distribution fixed and leaves only its scale free – the identification argument those papers make. Implemented for `model_name = "NTN"` only: for a half-normal u , $h(z)|N(0, \sigma^2)|$ is $|N(0, (h\sigma)^2)|$, so `uhet` already fits that model exactly and a separate option would be two names for one thing. Cannot be combined with `uhet` or `muhet`, since letting those vary separately would undo the restriction. The reported scale coefficients are δ ; `Zu.(Intercept)` and `Zmu.(Intercept)` are the scale and pre-truncation mean of the common draw.

<code>start_from</code>	Optionally, a previously fitted "sfareg" object to take starting values from. Parameters are matched by name , not by position: "NE" reports <code>sigv/sigu</code> where "NHN" reports <code>lambda/sigma</code> , so a positional copy would slide a variance ratio into a standard deviation. Anything unmatched keeps the start the package computes for it, and a fit sharing nothing but the frontier coefficients carries only those. Fitting a simpler model first and cascading is the documented remedy for the models with awkward likelihoods: over forty samples for "NG" seeded from "NE", the seeded fit was never worse, indistinguishable on thirty-nine, and worth 176.7 log-likelihood points on one. The rescue is rare and large rather than common and small, which is what makes it worth doing: the sample it rescues is one where the default start converges quietly to a much worse optimum, with nothing in the returned object to say so.
<code>Nsim</code>	Total number of simulation draws per observation for the simulated-maximum-likelihood models ("NLN", "NW"); ignored by every closed-form model. "auto" (the default) uses <code>max(200, ceiling(3*sqrt(n)))</code> . Half the draws are taken from each of the two proposals described under 'Details'. Simulated ML is consistent only if the draw count grows with the sample size – at a fixed <code>Nsim</code> the simulation bias does not vanish and every parameter except the frontier slopes converges to the wrong value. The previous fixed default of 100 did exactly that: at $n = 3000$ it returned (0.362, 0.821, 1.284, 0.328) against a truth of (0.3, 1.0, 1.5, 0.5) for "NW". A numeric value below the automatic floor warns. Earlier versions warned that "NLN" needed far more draws than "NW"; that was a property of the old integration scheme, not of the model. Both now behave alike: at $n = 3000$, measured against quadrature at the true parameters, total simulation error is 3.4 log-likelihood units for "NLN" and 3.0 for "NW" at the default, falling to 0.07 and 0.06 at <code>Nsim = 6400</code> . The comparable figure for the old scheme was 226.8.
<code>rand.psoptim</code>	Integer. seed for replication of <code>psoptim</code> . Default to NULL.
<code>keep_objective</code>	Logical. Store the likelihood function on the fitted object, so that <code>sfa_diagnostics</code> can compute the gradient at the optimum and draw likelihood slices. Defaults to FALSE: a closure carries its enclosing environment, so a fit saved with one serializes the estimation data too – on a 200-observation example the object grows from about 38 KB to 1.7 MB. Everything else <code>sfa_diagnostics()</code> reports works without it.
<code>estimator</code>	"mle" (default), or "cols"/"mols" – two names for the same thing, see 'Details' – which select the modified ordinary least squares moment estimator instead of maximum likelihood: OLS slopes are kept as they are, the second and third central moments of the OLS residuals are inverted for the scale parameters, and the

intercept is shifted up by the implied $E[u]$. Closed form, no optimizer, deterministic. Implemented for "NHN", "NE" and "NG"; other models error, because the moment inversion is distribution-specific. See 'Details'.

z_link	Scale on which the variance-determinant linear predictor is placed, for the <code>_Z</code> models. "sd" (the default, and sfm's historical convention) sets $\sigma_u = \exp(z'\delta)$; "var" sets $\sigma_u = \sqrt{\exp(z'\delta)}$, which is what <code>psfm</code> and the competing packages use. The two are reparameterizations of the same model – they reach the same maximised log-likelihood and the same <code>marginal_effects</code> – but δ itself differs by a factor of two, so a δ from sfm is not comparable with one from <code>psfm</code> or from <code>sfaR</code> unless the same convention is used. Set <code>z_link = "var"</code> to put them on one footing. See 'Details'.
vhet	One-sided formula for the determinants of the <i>noise</i> scale, e.g. <code>vhet = ~ z1 + z2</code> , giving $\sigma_v = \exp(z'_v\delta_v)$ under the SD link. NULL (the default) leaves σ_v constant. Available for "NHN", "NHN_Z", "NE", "NE_Z" and "NTN".
uhet	One-sided formula for the determinants of σ_u . This is the same specification as the <code> z</code> segment of formula, and it is an error to give both; <code>uhet</code> exists so that "NTN", which takes no pipe segment, can still have a heteroskedastic σ_u .
muhet	One-sided formula for the determinants of the <i>pre-truncation mean</i> of u , $\mu = z'_\mu\delta_\mu$. This is the Battese and Coelli (1995) inefficiency-effects model and requires <code>model_name = "NTN"</code> , the only family in which μ exists. Unlike the two scale blocks μ carries no link, since it may take either sign.
sim_type	Sequence used to build the simulated-ML draws, for the models fitted by simulation ("NLN", "NW"). "halton" (default) reproduces previous behaviour. "sobol" accepts <code>sim_scrambling</code> 1–3, giving the scrambled sequence of Bhat (2003). "torus" is the Korobov sequence and accepts <code>sim_prime</code> . "uniform" is pseudorandom, present as a baseline to measure against rather than as a recommendation. See 'Details'.
antithetics	Logical. If TRUE, half the draws are taken from the sequence and the other half created as their mirror images (Hammersley and Morton 1956). Default FALSE.
sim_burn	Number of leading elements of the sequence discarded. Default NULL, meaning 1000. The initial elements of different sequences are correlated with one another and discarding them removes it.
sim_scrambling	Integer 0–3, <code>sim_type = "sobol"</code> only, passed to <code>randtoolbox::sobol</code> . Non-zero gives a scrambled sequence.
sim_prime	Prime for <code>sim_type = "torus"</code> . Default NULL uses the generator's own choice.
sim_seed	Seed for the draws, for reproducibility. The caller's random-number stream is restored afterwards.
cols_boot	Integer. Number of nonparametric bootstrap resamples used to obtain standard errors under <code>estimator = "cols"</code> . Zero (the default) skips the bootstrap, in which case only the OLS slope standard errors are reported and the scale parameters and corrected intercept carry NA. Ignored when <code>estimator = "mle"</code> .
rand.cols	Integer. Seed for the COLS bootstrap, for reproducibility. The caller's random-number stream is restored afterwards. Default NULL.

Details

The options include the Normal-Half Normal (NHN), Normal-exponential (NE), Student's t-Half t (THT), the Normal-Truncated Normal (NTN), the Normal-Uniform (NU), the Normal-Generalized Exponential (NGE), the Normal-Lognormal (NLN), the Normal-Weibull (NW), the Student's t-half normal (tHN) and the Normal-Truncated Skew-Laplace (TSL). NHN_Z and NE_Z are extensions for the NHN and NE models that allow for modeling the u-component of those models with determinants of inefficiency.

Modified ordinary least squares (estimator = "cols" or "mols"). The moment estimator of Olson, Schmidt and Waldman (1980, *Journal of Econometrics*), standardly called MOLS. Note the name: this is *not* Winsten's corrected OLS, which shifts the intercept by the largest residual and estimates no variance parameters at all. The package has shipped this estimator under the spelling "cols" since it was introduced, so that spelling continues to work and is not deprecated; "mols" is accepted as an exact synonym so the method can be found under its usual name. OLS is consistent for the *slopes* of a composed-error frontier whatever the one-sided distribution; only the intercept is biased, by $E[u]$, because the composed error does not have zero mean. COLS therefore keeps the OLS slopes, inverts the central moments of the OLS residuals for the scale parameters, and shifts the intercept up by the implied $E[u]$. Writing m_2, m_3 for the second and third central moments of $e = v - u$: for "NHN", $m_3 = \sigma_u^3 \sqrt{2/\pi}(1 - 4/\pi)$ and $m_2 = \sigma_v^2 + \sigma_u^2(1 - 2/\pi)$ with $E[u] = \sigma_u \sqrt{2/\pi}$; for "NE", $m_3 = -2\sigma_u^3$ and $m_2 = \sigma_v^2 + \sigma_u^2$ with $E[u] = \sigma_u$; for "NG", from the third and fourth cumulants, $\sigma_u = -k_4/(3k_3)$ and $\mu = -k_3/(2\sigma_u^3)$ with $E[u] = \mu\sigma_u$.

No optimizer runs and the answer is deterministic, which makes COLS a natural robustness check against a maximum-likelihood fit that may have settled at a local optimum. It is less efficient than ML when the distributional assumption holds, and the "NG" inversion in particular leans on a fourth cumulant and is noticeably noisier than the other two at moderate sample sizes.

Wrong skew under COLS. A production frontier implies $m_3 < 0$. When a sample comes out with $m_3 \geq 0$ the moment equations have no admissible solution: `sfm()` warns, reports `sigu` as 0 with the whole residual variance assigned to `sigv`, and returns no efficiency predictions. Read that as *no evidence of inefficiency in these data*, not as an estimate of zero. This is the Type I failure of Olson, Schmidt and Waldman and is common in small samples; `$wrong_skew` and `$residual_moments` record it.

NR is not NHN. "NR" places a Rayleigh distribution on u , not a half-normal, and the two are separate families rather than reparameterizations of one another: the Rayleigh density carries an extra factor of u , and no rescaling removes it. The practical consequence is that the standardized skewness each contributes to the composed error is a different constant, -0.631 against the half-normal's -0.995 , so a sample generated under one cannot be fitted by the other. σ_u is reported on the second-raw-moment convention $E[u^2] = \sigma_u^2$, matching "NHN" and making the two directly comparable; the Rayleigh scale is then $\sigma_u/\sqrt{2}$, with $E[u] = \sigma_u \sqrt{\pi}/2$ and $Var(u) = (1 - \pi/4)\sigma_u^2$. Starting values come from inverting these moments, since the Rayleigh skewness is a constant and so m_3 identifies $Var(u)$ outright. Use `data_gen_cs()`'s `y_pcs_r` column to test it, not `y_pcs`.

tHN is not THT. In THT (Tancredi 2002) a single scale mixture is shared by both error components, so the noise and the inefficiency are both Student's t with the same degrees of freedom, and that shared mixture is what makes the composed error a closed-form skew-t. In tHN the noise is Student's t and the inefficiency is half-normal, drawn *independently*, so the two components have different tail behaviour, there is no closed form, and the density is the convolution $f(e) = \int_0^\infty f_v(e + u)f_u(u)du$ evaluated by Gauss-Legendre quadrature. tHN is the heavy-tailed-noise model, which is what makes it the natural parametric comparison for the density-power robust estimators (`robust =`

"mlqe", "psi", "mdpd"); THT cannot play that role because its inefficiency term is heavy-tailed too. Parameters are reported as (sigv, sigu, nu), the conventional order, *not* THT's inverted (sigu, sigv, a).

Two properties of thn are worth knowing before using it. First, the degrees of freedom are weakly identified: on data simulated from the model at $n = 1000$ with a true $\nu = 5$, the profile log-likelihood varies by only about 0.24 across ν from 10 to 100 and peaks near 20, so a single reported ν should not be taken at face value. Profile over a grid of fixed ν and report the path. Because of that flat ridge sfm() always fits thn from several widely separated starting values, keeps the best, and records the outcome in thn_starts; it warns when they reach different optima. Second, on real data σ_u can collapse onto zero, the heavy noise tail absorbing the entire one-sided component and leaving mean predicted efficiency near one. That is a property of the model rather than a numerical failure, so it is reported rather than suppressed: sfm() warns and sets thn_sigma_u_at_bound. Fits are also substantially slower than the closed-form models, since every likelihood evaluation is a quadrature whose node count grows with σ_u/σ_v .

"NU" assumes $u \sim U(0, \theta)$ (Li 1996; Nguyen 2010), giving the closed-form composed density $f(e) = \theta^{-1}[\Phi((e + \theta)/\sigma_v) - \Phi(e/\sigma_v)]$. The reported theta is the upper bound of the inefficiency support, so predicted inefficiency is bounded by construction. "NGE" assumes a generalized exponential $F(u) = (1 - e^{-\lambda u})^2$, whose composed density is a difference of two exponentially-tilted normal tail terms; it nests the exponential model and permits a non-monotonic inefficiency density. NGE reports sigu on the same scale as NE (the exponential mean, $\lambda = 1/\sigma_u$) so the two are directly comparable.

The variance-determinant scale. sfm's _Z models place $z'\delta$ on the standard deviation, $\sigma_u = \exp(z'\delta)$, while psfm's TRE_Z/GTRE_Z place it on the variance, $\sigma_u = \sqrt{\exp(z'\delta)}$, as do the competing packages. Since $\exp(\eta) = \exp(\eta/2)^2$, fitting the same data under the two conventions returns the same σ_u , the same log-likelihood and the same marginal effects, but δ under the SD link is exactly **half** δ under the variance link. Reading a δ from one family as though it came from the other therefore doubles or halves every reported effect. z_link exists so the two can be aligned; the default preserves sfm's own convention so existing results do not move.

Heteroskedasticity in more than one component. vhet, uhet and muhet let σ_v , σ_u and (for "NTN") the pre-truncation mean μ each depend on their own covariates, in any combination. They are named formulas rather than further pipe segments deliberately: pipe *position* already carries different meanings in different model families – the second segment is σ_w in ttsfm but σ_h in psfm's GTRE_Z – and a fourth position would not be readable. The existing | z segment keeps its meaning, so nothing that worked before changes.

Two reasons to reach for vhet in particular. Ignored heteroskedasticity in v does not merely cost efficiency: Caudill, Ford and Gropper (1995) show that it biases the estimated frontier, because the composed error's mean depends on the scales. And a σ_v that varies with observed characteristics is one of the few specifications that can absorb apparent wrong skewness without forcing σ_u onto the zero boundary. muhet with model_name = "NTN" is the Battese and Coelli (1995) inefficiency-effects model, in which covariates shift the location of inefficiency rather than its dispersion; that is usually the model wanted when the question is "what makes firms inefficient", since δ_μ then has a sign that reads directly.

These fits carry every positive quantity on the log scale, so the parameter vector is unconstrained and no variance can be driven onto a boundary by the optimizer; that is also why this path uses nlminb followed by an unbounded BFGS rather than the bounded four-stage scaffold the other models use. Coefficients are reported prefixed by block – Zv., Zu., Zmu. – since the same covariate may appear in more than one. marginal_effects reads the σ_u block of these fits with no extra

arguments. `estimator = "cols"` and the robust divergences are moment-based and homoskedastic respectively, so neither combines with these arguments; both refuse rather than silently ignoring them.

Simulated-ML draws. "NLN" and "NW" have no closed-form composed density and are fitted by maximum simulated likelihood, averaging over draws built by `sim_type`. Two properties are enforced whatever the sequence. Each observation receives its own *contiguous block* of the sequence rather than a strided subsequence: a column-major fill hands observation i the elements $h_i, h_{i+n}, \dots$ of a van der Corput sequence, which is not equidistributed – at $n = 500$ and $Nsim = 100$ the first observation's block spanned only $[0.50, 0.75]$, integrating over a quarter of the inefficiency distribution. And the draws are clamped away from 0 and 1, since `qnorm()` of an exact endpoint is infinite.

Blocking by observation is also what makes the draws worth using: Train (2002, ch. 9) attributes the advantage of a low-discrepancy sequence to its coverage *and* to the negative correlation it induces across observations, and the second only arises when observations receive different blocks. On draw counts, Bhat (2001) found 100 Halton draws more precise than 1000 pseudorandom ones, with 125 Halton draws giving half the simulation error of 1000 random draws; `Nsim = "auto"` scales the count with n and warns if a supplied count is too small.

"TSL" assumes a truncated skew-Laplace inefficiency (Wang 2012, eq. 3.7; the distribution is due to Aryal and Rao 2005, and the model was published as Nguyen, Gupta and Wang 2014), with density $f(u) = \frac{1+\lambda}{\sigma_u(2\lambda+1)} [2e^{-u/\sigma_u} - e^{-(1+\lambda)u/\sigma_u}]$ for $u \geq 0$. This is a *signed* mixture of two exponentials – the second enters negatively – so it nests the exponential model as $\lambda \rightarrow 0$ while allowing a non-monotonic inefficiency density, and the composed density is the corresponding signed combination of two normal-exponential terms. It is evaluated as a difference in log space rather than by subtracting the raw terms: both carry $\sigma_v^2/(2\sigma_u^2)$, which overflows once σ_u becomes small relative to σ_v , a region the optimizer routinely visits. `lambda` is a shape parameter and is the least sharply identified of the three. **This is a property of the family, not of the fit, and it is worth understanding before reporting `lambda` at all.** The shape parameter changes the standardised distribution remarkably little: $E[U]/\sigma_u$ equals 1 in both limits $\lambda \rightarrow 0$ and $\lambda \rightarrow \infty$, and its maximum over all λ is only $4 - 2\sqrt{2} \approx 1.1716$, attained at $\lambda = 1/\sqrt{2}$; the standardised skewness moves only from about 1.95 down to 1.79 and back. So the data identify the product $\sigma_u \times m(\lambda)$ far better than they identify σ_u and λ separately, and the two trade off against each other.

Measured on the package's own convergence design ($\sigma_u = 1$, $\lambda = 1.5$, 60 replications at each of $n = 500, 1000, 2000, 4000$), estimating one parameter with the other held at its true value:

estimated	log-MSE on log-n slope	sd at n = 4000
σ_u, λ free	-0.65	0.038
σ_u, λ fixed	-1.12	0.020
λ, σ_u free	-0.38 (median)	26.3
λ, σ_u fixed	-1.07 (median)	0.62

Each parameter attains the root- n rate when the other is known and fails it when the other is free, which is the signature of weak scale–shape separation rather than of an optimiser problem. λ 's dispersion falls fortyfold once σ_u is fixed.

What to do about it. Report the functionals rather than the primitives where you can: $E[U]$ converges at the root- n rate even with both parameters free (slope -1.03, sd 0.031 at $n = 4000$), and $E[e^{-U}]$ nearly so (-0.89). Both are what an efficiency analysis actually uses. If λ itself is

of interest, prefer profile-based inference over its reported t -value, which is a Wald statistic on a parameter whose sampling distribution is neither symmetric nor tightly concentrated.

"NLN" (u lognormal, reported as `sigu = sdlog` and `mu = meanlog`) and "NW" (u Weibull, reported as `sigu = scale` and `k = shape`) have no closed-form composed density and are estimated by *simulated* maximum likelihood. The integral $f(e) = \int_0^\infty f_u(u) \phi((e + u)/\sigma_v)/\sigma_v du$ is a product of two densities, and simulating from either one alone fails wherever the other is the narrow factor: draws from f_u miss the normal kernel for a very inefficient unit, while draws from the kernel miss a sharply peaked f_u . Half the draws are therefore taken from each, and combined by the balance heuristic of Veach and Guibas (1995), which weights every draw by the proposal likelier to have produced it. This needs no rule for choosing between the two and remains accurate in both regimes. The draws are generated once and reused at every optimizer iteration, so the objective is deterministic and reproducible; increase `Nsim` to reduce simulation error.

The efficiency predictor reuses the weights the likelihood computed, so `u_hat` is the Bayes rule under exactly the density that was maximised.

Outputs include $E[\exp(-u)|e]$ given by `exp_u_hat`, following Battese and Coelli (1988, JoE), where appropriate. As of this version `exp_u_hat` is also returned for NE, NTN, NU, NGE, NLN and NW, together with the Jondrow et al. (1982) point predictor `u_hat = E[u|e]`; previously the exponential and truncated-normal models returned no efficiency prediction at all.

Note on default iteration counts: `sfm()`'s optimization defaults (thousands of iterations) are much higher than `psfm()`'s (100/10/10). This is intentional: `sfm()`'s likelihoods are closed-form (no simulation), so each objective-function evaluation is cheap and many default iterations cost little; several of `psfm()`'s models require simulated maximum likelihood over Halton draws, where each evaluation is far more expensive. See [psfm's Details](#) for more.

Robust divergence-based estimation (`robust argument`). For `model_name = "NHN"`, `sfm()` can maximize a robust divergence objective instead of the ordinary log-likelihood, downweighting the influence of outlying observations:

- "mlqe": minimum L_q estimation (Lq, Ferrari and Yang 2010), power parameter c_{mlqe} ($q = 1 - c$). Objective $\sum (f(e_i)^c - 1)/c$.
- "psi": Psi-likelihood (Basu et al.-type divergence family), power parameter η . Objective $\sum f(e_i)^c/c - n I_{1+c}/(1 + c)$, where $I_{1+c} = \int f(e; \theta)^{1+c} de$ is evaluated numerically at the current parameter values on every optimizer iteration.
- "mdpd": minimum density power divergence (Basu et al. 1998), power parameter α . Objective $((1 + c)/c) \sum f(e_i)^c - n I_{1+c}$.

All three reduce to "mle" as their power parameter approaches 0. **Psi and MDPD produce identical parameter point estimates for the same power parameter** ($c = \eta = \alpha$) – the two objectives are a positive rescaling of one another, $MDPD(c) = (1 + c) \cdot \text{Psi}(c)$, which does not change the `argmax`; both are exposed for compatibility with the corresponding paper's three-method terminology, not because they differ numerically. MLQE is a genuinely distinct estimator.

Because these are M-estimators rather than ordinary MLE, the naive Hessian-inverse standard error (`optHessian = TRUE`'s usual output) is not statistically valid for them – it assumes the information-matrix equality, which these objectives do not satisfy in general. When `robust != "mle"`, `sfm()` instead reports a sandwich-form standard error, $A^{-1}BA^{-1}$, where A is the Hessian of the summed robust objective (as already computed for the point estimate) and $B = \sum_i g_i g_i'$ is built from the per-observation gradient of that objective (obtained via numerical differentiation, [jacobian](#)). This requires `optHessian = TRUE`; `std.errors/t.values` are NA otherwise, same as for ordinary MLE.

Formulas verified against an independently audited reference implementation (cross-checked by direct numerical integration and finite differences), including the sandwich SE's own verification (the $c \rightarrow 0$ limit against the ordinary MLE Hessian SE, and a Monte Carlo check against the empirical spread of repeated fits). Currently only implemented for `model_name = "NHN"` – other models will error clearly if `robust != "mle"` is requested.

Value

An object of class `"sfareg"` containing the following components:

<code>out</code>	A matrix with parameter estimates, standard errors, and t-values.
<code>opt</code>	A list containing the optimization results from the final optimization procedure.
<code>total_time</code>	The total computation time for model estimation.
<code>start_v</code>	The starting values used in the optimization.
<code>model_name</code>	The name of the stochastic frontier model estimated.
<code>formula</code>	The formula used in the model specification.
<code>exp_u_hat</code>	Predicted technical efficiency, $E[\exp(-u) e]$. Available for models: NHN, NHN_Z, NR, NG, NNAK, NE, NTN, NU, NGE, NLN, NW, THT and tHN.
<code>u_hat</code>	Predicted inefficiency, $E[u e]$ (Jondrow et al. 1982). Available for NE, NTN, NU, NGE, NLN, NW, THT and tHN.
<code>wrong_skew, sigma_u_at_bound, residual_m3</code>	Wrong-skewness diagnostics. <code>residual_m3</code> is the third central moment of the OLS residuals; <code>wrong_skew</code> is TRUE when that is non-negative, which for a production frontier means the moment equations admit no interior solution; <code>sigma_u_at_bound</code> is TRUE when the fitted one-sided scale has collapsed to the zero boundary. When both are TRUE the boundary <i>is</i> the maximum likelihood estimate – the Type I failure of Olson, Schmidt and Waldman (1980), studied by Waldman (1982) – and not a numerical failure, so <code>sfm</code> reports it rather than bounding σ_u away from zero. Read it as no evidence of inefficiency in these data, and treat the efficiency scores as uninformative. A collapse with correctly skewed residuals would be a different matter and does not occur in testing.
<code>thn_sigma_u_at_bound</code>	Logical, tHN only. TRUE when the fitted σ_u collapsed onto zero, so the model reports essentially no inefficiency. Accompanied by a warning. Treat the efficiency scores as uninformative when this is set.
<code>thn_starts</code>	List, tHN only. Multi-start diagnostic: how many starting values were tried and converged, the log-likelihood reached from each, and <code>n_distinct</code> , the number of separate optima found. <code>n_distinct > 1</code> means the likelihood surface is multimodal for these data and the single returned fit is only the best of those found.
<code>estimator</code>	"cols" for a corrected-OLS fit; absent for maximum-likelihood fits.
<code>wrong_skew</code>	Logical, <code>estimator = "cols"</code> only. TRUE when the OLS residuals were skewed the wrong way and the moment equations had no admissible solution.
<code>residual_moments</code>	The second, third and fourth central moments of the OLS residuals, <code>estimator = "cols"</code> only.

cols_boot_draws	The matrix of bootstrap replicates (cols_boot rows, one column per parameter) when cols_boot > 0; NULL otherwise.
ng_starts	Returned by NG only. A record of the starting-value search: how many candidates were built (n_tried), how many were polished before choosing (n_polished), the log-likelihood at each candidate (loglik_at_start), and the best value reached (best). NULL when start_val was supplied. The normal-gamma likelihood determines $E[u] = \mu\sigma_u$ far better than it determines either factor – the weak identification Ritter and Simar (1997) describe – so the candidates hold $E[u]$ at a moment estimate and sweep the shape along that ridge rather than searching across it. Inspect this if a normal-gamma fit looks implausible: widely differing log-likelihoods among the candidates indicate the surface is more awkward than usual for that data set.
sd_exp_u_hat	Standard deviation of exp(-u) given the residual, following Tancredi (2002, section 2.2). Returned by THT only. It is the quantity that distinguishes the skew-t frontier from the half-normal one: for a large <i>positive</i> residual the half-normal model concentrates exp(-u) at 1 and reports near-zero uncertainty, while the skew-t model treats the point as an outlier, widens this spread and does not push the efficiency estimate to 1.
med_u_hat	Predicted technical efficiency (median values). Available only for the NHN model.
coefficients	A vector of estimated parameters.
std.errors	A vector of standard errors for the estimated parameters (NA if optHessian = FALSE). Sandwich-form (not the ordinary Hessian-inverse form) when robust != "mle" – see Details.
t.values	A vector of t-values for the estimated parameters (NA if optHessian = FALSE).
call	The matched call.
het	Only present when vhet, uheter or muhet was given: the family fitted, the link used, the sizes of the four parameter blocks, the formulas as supplied, and (for "NTN") the fitted per-observation μ .
sigma_u, sigma_v	Only present for a heteroskedastic fit: the fitted per-observation scales.
v_spec, mu_spec	Only present for a heteroskedastic fit: the noise and pre-truncation-mean blocks, in the same shape as z_spec.
robust	Only present for model_name = "NHN": the robust argument used to fit this model ("mle", "mlqe", "psi", or "mdpd").
robust_c	Only present for model_name = "NHN": the power parameter (c_mlqe/eta/alpha, whichever applies) used, or NA when robust = "mle".

Note

Standard errors require optHessian set to TRUE. When robust != "mle" (model_name = "NHN" only), a sandwich-form SE is reported instead of the ordinary Hessian-inverse one – see Details.

Author(s)

David H. Bernstein and Alexander Stead

See Also

[psfm](#) for panel models, [zsfm](#) for zero-inefficiency models, [ttsfm](#) for two-tier models, [data_gen_cs](#) for simulating data with known true parameters, and [sfareg-methods](#) for extracting results.

Examples

```
library(sfa)

cs_data_trial <- data_gen_cs(N= 1000, rand = 1, sig_u = 0.3, sig_v = 0.3,
  cons = 0.5, beta1 = 0.5, beta2 = 0.5, a = 4, mu = 1)

cs.nhnz <- sfm(formula = y_pcs_z ~ x1 +x2| z, model_name = "NHN_Z",
  data = cs_data_trial, PSopt = TRUE)

## Robust divergence-based estimation (NHN only): downweights the influence
## of outlying observations relative to ordinary MLE. std.errors/t.values
## are NA for these methods -- see Details.
cs.nhn.mlqe <- sfm(formula = y_pcs ~ x1 + x2, model_name = "NHN",
  data = cs_data_trial, robust = "mlqe", c_mlqe = 0.20)
cs.nhn.psi <- sfm(formula = y_pcs ~ x1 + x2, model_name = "NHN",
  data = cs_data_trial, robust = "psi", eta = 0.20)

## Heteroskedasticity in the noise as well as in the inefficiency: `| z`
## still drives sigma_u, `vhet` drives sigma_v. Coefficients come back
## prefixed by block (Zu., Zv.).
cs.het <- sfm(formula = y_pcs_z ~ x1 + x2 | z, model_name = "NHN_Z",
  data = cs_data_trial, vhet = ~ z)

## Battese and Coelli (1995): covariates shift the LOCATION of inefficiency
## rather than its dispersion. Requires the truncated-normal family.
cs.bc95 <- sfm(formula = y_pcs ~ x1 + x2, model_name = "NTN",
  data = cs_data_trial, muhet = ~ z)
```

sfma

Average over inefficiency distributions instead of choosing one

Description

Fits several inefficiency distributions to the same data and combines their firm-level inefficiency predictions with data-driven weights. Where [TIC](#) and [vuong](#) answer "which distribution?", this answers what to do when the honest reply is "the data does not say" – selecting one specification and then reporting its efficiencies understates the uncertainty, because the standard errors condition on a choice that was itself estimated.

Usage

```
sfma(formula, data, models = c("NHN", "NE", "NTN", "NR", "NGE"),
  weights = c("sfma", "tic", "aic", "bic", "equal"),
  inefdec = TRUE, quiet = FALSE, ...)
```

Arguments

formula, data, inefdec, ...	Passed to <code>sfm</code> for each candidate.
models	Two or more <code>model_names</code> to average over. Candidates that fail to fit, or that report no $E[u \mid \varepsilon]$, are dropped with a message.
weights	How to weight the candidates. See ‘Details’.
quiet	Suppress the messages about dropped candidates.

Details

The weighting schemes.

"sfma" is the criterion of Parmeter, Wan and Zhang (their eq. 13),

$$C(w) = \|\hat{\rho}(w) - \hat{\rho}_{full}\|^2 + n^{1/2} \log(n) k'w,$$

minimised over the unit simplex. The unobservable target is replaced by the prediction from the most general candidate, exactly as Mallows' C_p replaces an unknown σ^2 with the full model's estimate; without the penalty all weight would go to the largest candidate by construction. Their Theorem 1 says weights on models that nest the truth go to zero, and they do.

"tic", "aic" and "bic" are smoothed information-criterion weights in Buckland *et al.*'s (1997) form, $w_s \propto \exp(-IC_s/2)$. "tic" is the interesting one, because unlike AIC it does not assume the candidate is correctly specified – which is the premise of averaging in the first place. "equal" is the naive average, useful as a baseline.

"sfma" **needs a unique most general candidate, and refuses without one.** If every candidate has the same number of parameters the penalty $n^{1/2} \log(n) k'w$ is identical across them, so it drops out as a constant and the criterion collapses to "be as close as possible to $\hat{\rho}_{full}$ " – minimised by putting weight 1 on whichever model was nominated as full. Measured on exponentially-distributed inefficiency with candidates "NHN", "NE" and "NR" (all with 5 parameters), that would have placed 0.879 on "NHN", the wrong distribution, where "tic" gave "NE" 0.934. Rather than break the tie silently, sfma errors and points at "tic". Including a genuinely more general candidate such as "NTN" makes the criterion well posed.

Reading the output. `print` reports the per-firm spread of efficiency across candidates. That spread is the reason to average: if the candidates agree, model choice was not costing anything and any single fit would have done; if they disagree, a single-model standard error is not describing the uncertainty a reader cares about.

Implementation. The weight problem is a convex quadratic program on the simplex, solved by projected gradient with an exact simplex projection (Duchi *et al.* 2008), so a weight that should be zero is exactly zero. It is checked against brute-force enumeration of the simplex in the tests rather than against itself.

Value

An object of class "sfma" with components `weights`, `u_hat` (the averaged $E[u \mid \varepsilon]$), `efficiency`, `u_by_model` (the $n \times S$ matrix of per-candidate predictions), `fits`, `k`, `logLik` and, for `weights = "sfma"`, `criterion`.

References

- Buckland, S.T., Burnham, K.P. and Augustin, N.H. (1997). Model selection: an integral part of inference. *Biometrics*, 53(2), 603–618.
- Duchi, J., Shalev-Shwartz, S., Singer, Y. and Chandra, T. (2008). Efficient projections onto the l_1 -ball for learning in high dimensions. *ICML*.
- Huang, C.J. and Lai, H.-p. (2012). Estimation of stochastic frontier models based on multimodel inference. *Journal of Productivity Analysis*, 38(3), 273–284.
- Parmeter, C.F., Wan, A.T.K. and Zhang, X. (2019). Model averaging estimators for the stochastic frontier model. *Journal of Productivity Analysis*, 51(1), 91–103.

See Also

[TIC](#), [vuong](#), [spec_test](#), [sfm](#)

Examples

```
set.seed(3)
n <- 300
x1 <- rnorm(n); x2 <- rnorm(n)
## True inefficiency is exponential.
d <- data.frame(y = 1 + 0.5 * x1 + 0.5 * x2 +
                rnorm(n, 0, 0.4) - rexp(n, 1),
                x1 = x1, x2 = x2)

## Information-criterion weights: NE should earn most of the weight.
a <- sfma(y ~ x1 + x2, data = d, models = c("NHN", "NE", "NR"),
          weights = "tic", quiet = TRUE)
a
a$weights

## The Parmeter-Wan-Zhang criterion needs a uniquely largest candidate.
sfma(y ~ x1 + x2, data = d, models = c("NHN", "NE", "NTN"),
     weights = "sfma", quiet = TRUE)
```

simulation_se

Simulation Standard Errors for Simulated-ML Fits

Description

Reports how much of a fitted parameter's uncertainty is *simulation* noise rather than sampling noise, by refitting the same data with independent randomizations of the draw sequence.

Usage

```
simulation_se(object, K = 10, seeds = NULL, quiet = FALSE,
             envir = parent.frame())
```

Arguments

object	An "sfareg" fit from a simulated-ML model: <code>sfm</code> with "NLN" or "NW", <code>psfm</code> with "GTRE", "TRE", "TRE_Z" or "GTRE_Z", or <code>selsfm</code> .
K	Number of refits, at least 2.
seeds	Optional integer vector of randomization seeds; its length overrides K.
quiet	Suppress the per-refit progress messages.
envir	Environment in which to re-evaluate the stored call. The default is usually right; pass it explicitly if the data cannot otherwise be found.

Details

Simulated maximum likelihood replaces an integral with an average over draws. A standard error from the Hessian answers “how much would this estimate move in a new sample”, and says nothing about “how much would it move with the *same* sample and different draws”. Randomizing the sequence makes the second question answerable: refit K times with independent shifts and take the standard deviation across refits.

This is possible only because the draws are randomized by a uniform shift. An unrandomized Halton lattice returns the same answer every time, so the diagnostic would report exactly zero and mean nothing by it.

Reading the result. `ratio` is the share of the reported standard error that comes from having simulated the integral. A large value is an argument for raising `Nsim`, not for collecting more data. On the package’s reference design ("NW", $n = 400$, `Nsim = 200`) the frontier slopes come back at about 0.5% while the variance and shape parameters are near 8% – which independently reproduces the documented behaviour that too few draws move everything except the slopes.

Quadrupling `Nsim` on that design cuts the simulation standard errors by roughly a factor of six, faster than the factor of two that independent pseudorandom draws would give. That is the low-discrepancy sequence earning its place.

Value

A data frame with one row per parameter: `estimate`, `sampling_se` (from the Hessian), `simulation_se` (across refits) and their ratio. Attributes `n_refits` and `seed_arg` record how many refits succeeded and which argument was varied.

See Also

[sfa_diagnostics](#) for convergence diagnostics on a single fit.

Examples

```
## Kept deliberately small: every refit is a full simulated-ML fit, so the
## cost is K + 1 of them. K = 3 is the smallest that still shows the spread.
d <- data_gen_cs(N = 200, rand = 1, sig_u = 1, sig_v = 0.3, cons = 0.5,
  beta1 = 0.5, beta2 = 0.5, a = 1, mu = 0.5)
fit <- sfm(y_pcs_wb ~ x1 + x2, model_name = "NW", data = d, Nsim = 200)
simulation_se(fit, K = 3, quiet = TRUE)
```

skewness_test

*Formal tests for wrong skewness in the OLS residuals***Description**

Tests whether the OLS residuals of a stochastic frontier model are skewed in the direction the model implies. A production frontier has $e = v - u$ with $u \geq 0$, so the composed error is **negatively** skewed; residuals skewed the other way mean the maximum likelihood estimate of σ_u is exactly zero and every efficiency score is one.

Usage

```
skewness_test(object, test = c("agostino", "coelli"),
              alternative = c("auto", "less", "greater", "two.sided"))
```

Arguments

object	An object of class "sfareg" from sfm , or a numeric vector of residuals. From a fit, the OLS residuals recorded at fit time are used – not the composed residuals of the fitted model, which are not what either test is defined on.
test	"agostino" (the default) is D'Agostino's (1970) skewness test, which is the test Schmidt and Lin (1984) appeal to. "coelli" is Coelli's (1995) <i>M3T</i> . See 'Details' for why the default is the former.
alternative	"auto" (the default) runs the one-sided test the model implies: a production frontier predicts negative skew, so the null of zero skew is tested against $m_3 < 0$. The other three force a direction.

Details

What the tests are. Both start from the third central moment of the OLS residuals. Coelli's *M3T* = $m_3 / \sqrt{6m_2^3/n}$ is asymptotically standard normal under the null of zero skewness. D'Agostino's transforms $\sqrt{b_1} = m_3/m_2^{3/2}$ to approximate normality in finite samples rather than leaning on the asymptotic variance $6/n$.

Why "agostino" is the default. It holds its size where the asymptotic form does not. Measured here, one-sided at a nominal 5% over 4,000 replications of symmetric residuals:

n	25	50	100	400
coelli	0.031	0.037	0.043	0.050
agostino	0.045	0.046	0.046	0.051

Coelli's is conservative below about $n = 100$; the two agree by $n = 400$. The default also matches **sfaR**'s. The implementation agrees with `moments::agostino.test` to within 10^{-10} .

How to read the result. A small p-value under the default alternative is evidence *for* the frontier specification: the residuals are skewed the way inefficiency makes them. A large p-value is not evidence of a problem by itself – it may simply mean there is little inefficiency to detect, or too

little data to detect it. What matters is the combination with `wrong_skew`: residuals skewed the wrong way *and* a fit reporting `$sigma_u_at_bound` is the Type I failure of Olson, Schmidt and Waldman (1980), in which case the boundary really is the maximum likelihood estimate (Waldman 1982) and the efficiency scores carry no information. See [sfm](#).

Value

An object of class "htest", so it prints like any other R test. Alongside the usual statistic, p.value and estimate (the sample skewness $\sqrt{b_1}$) it carries `m3`, `nobs`, and `wrong_skew` – the last being TRUE when the third central moment is non-negative, matching `sfm`'s own `$wrong_skew`.

References

- Coelli, T. (1995). Estimators and hypothesis tests for a stochastic frontier function: a Monte Carlo analysis. *Journal of Productivity Analysis*, 6(3), 247–268.
- D'Agostino, R.B. (1970). Transformation to normality of the null distribution of g_1 . *Biometrika*, 57(3), 679–681.
- Olson, J.A., Schmidt, P. and Waldman, D.M. (1980). A Monte Carlo study of estimators of stochastic frontier production functions. *Journal of Econometrics*, 13(1), 67–82.
- Schmidt, P. and Lin, T.-F. (1984). Simple tests of alternative specifications in stochastic frontier models. *Journal of Econometrics*, 24(3), 349–361.
- Waldman, D.M. (1982). A stationary point for the stochastic frontier likelihood. *Journal of Econometrics*, 18(2), 275–279.

See Also

[moment_range](#) and [spec_test](#) for the Type II counterpart, [sfm](#), [sfa_diagnostics](#)

Examples

```
cs <- data_gen_cs(N = 200, rand = 1, sig_u = 0.3, sig_v = 0.4, cons = 0.5,
                 beta1 = 0.5, beta2 = 0.5, a = 5, mu = 0.1)
fit <- sfm(y_pcs_e ~ x1 + x2, model_name = "NE", data = cs)

skewness_test(fit)                # D'Agostino, one-sided
skewness_test(fit, test = "coelli") # Coelli's M3T

## Also works on a bare residual vector.
skewness_test(residuals(lm(y_pcs_e ~ x1 + x2, data = cs)))
```

spec_test

*Is this pair of distributions defensible?***Description**

Tests an assumed **pair** of distributions for the composed error $\varepsilon = v \pm u$ using only the skewness and excess kurtosis of the OLS residuals. It needs no frontier fit, so it can be run before estimating anything and cannot be contaminated by an optimiser that failed. `spec_test_all` runs all twelve supported pairs and ranks them.

Usage

```
spec_test(object, noise = c("normal", "logistic", "laplace", "uniform"),
          inefficiency = c("halfnormal", "exponential", "genexponential"),
          null = c("bootstrap", "asymptotic"),
          B = 999, level = 0.05, seed = NULL)

spec_test_all(object, level = 0.05)
```

Arguments

<code>object</code>	An "sfareg" fit, whose OLS residuals are used, or a numeric vector of residuals.
<code>noise</code>	Assumed distribution of v . Only its excess kurtosis enters, so any symmetric distribution with the same value behaves identically – testing for Logistic noise is indistinguishable from testing for $t(9)$.
<code>inefficiency</code>	Assumed distribution of u .
<code>null</code>	How to get the p-value. "bootstrap" (the default) simulates from the assumed pair; "asymptotic" uses the delta-method normal approximation, which is badly over-sized for heavy-tailed u – see 'Details'.
<code>B</code>	Bootstrap replications. Cheap here: nothing is fitted, so each replicate is n draws and eight moments.
<code>level</code>	Size used for the reported verdict.
<code>seed</code>	Optional seed for the bootstrap. The RNG state is saved and restored.

Details

What it tests. Write $R = \text{SNR}^2 / (1 + \text{SNR}^2)$ for the share of composed-error variance contributed by u . Papadopoulos and Parmeter (2021, eqs. 5 and 10) give

$$|\gamma_1(\varepsilon)| = \gamma_1(u) R^{3/2}, \quad \gamma_2(\varepsilon) = \gamma_2(v)(1 - R)^2 + \gamma_2(u)R^2.$$

With both distributions assumed, $\gamma_1(u)$, $\gamma_2(u)$ and $\gamma_2(v)$ are known constants, so the *skewness alone* identifies R and the kurtosis equation has nothing left to fit. The statistic is the gap,

$$\Gamma = \gamma_2(\varepsilon) - \gamma_2(v)(1 - R)^2 - \gamma_2(u)R^2,$$

which is zero when the assumed pair is right. Its variance comes from the delta method applied to the central moments (m_2, m_3, m_4) , which needs moments up to the eighth.

Why the default null is a bootstrap. The delta-method p-value is not reliable at usual sample sizes, and the failure is systematic rather than marginal. Rejection rates under *correct* specification, 300 replications per cell:

u	n	SNR	asymptotic	bootstrap
half-normal	2000	1	0.087	0.050
half-normal	2000	2	0.103	0.057
exponential	2000	1	0.207	0.077
exponential	500	2	0.357	0.083
exponential	2000	2	0.307	0.060

Across all 24 cells the asymptotic null rejects **18.7%** of the time at a nominal 5%; the bootstrap rejects **5.7%**. The cause is the eighth moment: for a heavy-tailed u it is estimated far too imprecisely at these sample sizes, and the variance comes out too small. The bootstrap studentises against the same statistic computed under the assumed pair, so it corrects the variance rather than merely relocating the statistic. Power is unaffected – against wrong pairs at $n = 2000$ it runs 0.93 to 1.00.

Type II failure. If $R > 1$ the residuals are more skewed than the assumed u can be – for instance $|\gamma_1| > 0.9953$ under a half-normal. The pair is then refuted by the skewness alone, and `spec_test` says so instead of reporting a p-value that would suggest the question was close.

Reading `spec_test_all`. These are not sequential tests: no one of them depends on another's outcome, so the procedure implies no multiplicity correction. But picking the largest of twelve p-values is a selection, and the ranking should be read as comparative evidence rather than as a significance level.

Relation to `skewness_test`. That asks whether the residuals are skewed the way inefficiency makes them – a test of whether there *is* inefficiency. This asks whether the assumed distributions are the right ones. They answer different questions and are worth running together.

Provenance. The statistic here is *reconstructed* from PP2021 equations (5) and (10); the Supplementary Appendix in which PP2023 derive it was not available. The reconstruction is verified two ways in the tests: Γ vanishes under correct specification across sixteen designs, and the tabulated constants $c_{u,1}$ and $c_{u,2}$ are reproduced exactly as $\gamma_1(u)^{-2/3}$ and $1/\gamma_2(u)$. PP2021's own equation (13) is deliberately not used, since PP2023 record that its H_k terms are missing squares.

Value

`spec_test` returns an object of class `c("sfa_spec_test", "htest")`, carrying statistic (the studentised Γ), p.value, Gamma, se, R, skewness, excess_kurtosis, type2_failure and reject. `spec_test_all` returns a data frame of all twelve pairs sorted by p-value.

References

Papadopoulos, A. and Parmeter, C.F. (2021). Type II failure and specification testing in the Stochastic Frontier Model. *European Journal of Operational Research*, 293(3), 990–1001.

Papadopoulos, A. and Parmeter, C.F. (2023). A specification test for the composed error term in the stochastic frontier model. *Economics Letters*, 233, 111390.

See Also

[moment_range](#) for the attainable ranges the same two equations imply, [skewness_test](#), [TIC](#), [sfm](#)

Examples

```
set.seed(2)
n <- 2000
e <- rnorm(n, 0, 0.6) - abs(rnorm(n, 0, 1)) # normal / half-normal

## The true pair is not rejected.
spec_test(e, noise = "normal", inefficiency = "halfnormal",
          B = 199, seed = 1)

## A wrong one is.
spec_test(e, noise = "laplace", inefficiency = "exponential",
          B = 199, seed = 1)

## All twelve, ranked.
spec_test_all(e)
```

summary.sfareg

Summary method for "sfareg" objects

Description

Summary of a stochastic frontier fit, as returned by [sfm](#), [psfm](#), [zsfm](#) or [ttsfm](#).

Usage

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

Arguments

object	An object of class "sfareg", as returned by sfm , psfm , zsfm or ttsfm .
...	Additional arguments passed to other methods

Details

Prints the call, the model name, and the estimated parameters with standard errors and t-values, together with the log-likelihood and the number of observations where the estimator provides them. Which parameters appear depends on the model: several report the $\lambda = \sigma_u / \sigma_v$, $\sigma = \sqrt{\sigma_u^2 + \sigma_v^2}$ reparameterization rather than the raw scale parameters.

Value

Invisibly returns its object argument unchanged, having printed the summary as a side effect.

Author(s)

David H. Bernstein

See Also

[print.sfareg, sfareg-methods](#)

Examples

```
library(sfa)

cs_data_trial <- data_gen_cs(N= 1000, rand = 1, sig_u = 0.3, sig_v = 0.3,
  cons = 0.5, beta1 = 0.5, beta2 = 0.5, a = 4, mu = 1)

cs.nhnz <- sfm(formula = y_pcs_z ~ x1 +x2| z, model_name = "NHN_Z",
  data = cs_data_trial, PSopt = TRUE)

summary(cs.nhnz)
```

TIC

Choosing between non-nested stochastic frontier specifications

Description

Model selection for stochastic frontier models when the candidates are **not nested** and none of them need be correctly specified. TIC is Takeuchi's information criterion, a version of [AIC](#) whose penalty is estimated from the data rather than assumed equal to the number of parameters. [vuong](#) is Vuong's pairwise test, which returns a p-value and is allowed to conclude that two specifications are indistinguishable.

Usage

```
TIC(object, detail = FALSE)
```

```
vuong(object1, object2, correction = c("none", "aic", "tic"), level = 0.05)
```

Arguments

object, object1, object2

Objects of class "sfareg", fitted with keep_objective = TRUE (see 'Details').

detail

If FALSE (the default) TIC returns a single number. If TRUE it returns a list also holding the log-likelihood, the parameter count, the estimated penalty and their ratio.

correction	Bias correction applied to the log-likelihood difference before it is standardised. "none" (the default) is the statistic whose $N(0, 1)$ limit Vuong established. "aic" subtracts the parameter counts and "tic" the Takeuchi penalties; both shift the mean of the statistic and neither has an established null distribution, so their p-values are flagged as indicative.
level	Two-sided size used for the reported critical value and the verdict.

Details

Why not AIC. `sfa` offers many `model_names` for the same data, and `AIC` assumes the fitted model is correctly specified – exactly the assumption in doubt when choosing between "NHN", "NE", "NG" and "NTN". Most such pairs are not nested either, so the ordinary likelihood ratio test has no chi-square limit.

What TIC does. Takeuchi's criterion replaces AIC's penalty d with $\text{tr}[H(\hat{\theta})I(\hat{\theta})^{-1}]$, where I is the sample Fisher information and H the outer product of the per-observation scores. Under correct specification the information matrix equality gives $H = I$, the trace collapses to d , and TIC equals AIC. The gap between penalty and `df` – reported as `ratio` under `detail = TRUE` – is therefore a readable diagnostic in its own right: a ratio near one says the distributional assumption is not doing visible damage.

What vuong does. It tests H_0 that the two models are equally close to the truth in the Kullback-Leibler sense. Neither model has to be correct. With m_i the difference of per-observation log-likelihoods, the statistic is $n^{-1/2} \sum_i m_i / \hat{\sigma}$, asymptotically standard normal. Because it is a test rather than a criterion it can return "neither", which is frequently the honest answer and is not available from AIC.

Requirement. Both need per-observation log-likelihoods, so the fits must be made with `keep_objective = TRUE`. This is supported by `sfm` (for "NHN", "NE", "NR", "NG", "NNAK", "THT", "NTN", "NHN_Z", "NE_Z", "NU", "NGE", "NLN", "NW", "tHN" and "TSL") and by `psfm`. `vuong` additionally requires both fits to use the same observations, which it checks by sample size and warns about when the data arguments differ.

Estimators that maximise nothing. `psfm`'s "GTRE_SEQ1", "GTRE_SEQ2" and "SSFE", and `ivsfm`'s "C2SLS", are moment-based or FE and have no log-likelihood, so no likelihood-based criterion applies to them and both functions refuse rather than returning a number built on NA.

Value

TIC returns a single numeric value, or with `detail = TRUE` a list with components `TIC`, `AIC`, `logLik`, `df`, `penalty` and `ratio`.

`vuong` returns an object of class "sfa_vuong" with components `statistic`, `p.value`, `n`, `lr`, `lr_adjusted`, `omega`, `correction`, `penalty`, `critical`, `level`, `models`, `favoured` and `exact_null`. Lower TIC is better; `favoured` is the name of the winning `model_name`, or "neither".

References

- Lai, H.-p. and Huang, C.J. (2010). Likelihood ratio tests for model selection of stochastic frontier models. *Journal of Productivity Analysis*, 34(1), 3–13.
- Takeuchi, K. (1976). Distribution of informational statistics and a criterion of model fitting. *Suri-Kagaku (Mathematical Sciences)*, 153, 12–18.

Vuong, Q.H. (1989). Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica*, 57(2), 307–333.

See Also

[sfm](#), [skewness_test](#), [sfa_diagnostics](#), [AIC](#)

Examples

```
set.seed(42)
n <- 400
x1 <- rnorm(n); x2 <- rnorm(n)
d <- data.frame(y = 1 + 0.5 * x1 + 0.5 * x2 +
                rnorm(n, 0, 0.4) - abs(rnorm(n)),
                x1 = x1, x2 = x2)

a <- sfm(y ~ x1 + x2, model_name = "NHN", data = d, keep_objective = TRUE)
b <- sfm(y ~ x1 + x2, model_name = "NE", data = d, keep_objective = TRUE)

## Lower is better, and comparable across non-nested models.
TIC(a)
TIC(b)

## Is the penalty near the parameter count? If so, TIC has little to add.
TIC(a, detail = TRUE)$ratio

## And a test, which may decline to pick a winner.
vuong(a, b)
```

ttsfm

Two-Tier Stochastic Frontier Model

Description

Code to estimate the two-tier stochastic frontier model (2TSF), which decomposes a symmetric error term into two one-sided components (e.g. buyer/seller informational rents) in addition to the usual two-sided noise term.

Usage

```
ttsfm(formula, model_name = c("TTNE", "TTHN", "TTNLS"),
data, z_link = c("sd", "var"), maxit.bobyqa = 80000,
maxit.psoptim = 1000, maxit.optim = 1000,
REPORT = 1, trace = 0, pgtol = 0, start_val = FALSE, PSopt = FALSE,
optHessian = TRUE, inefdec = TRUE, upper = NA,
Method = "L-BFGS-B", logit = TRUE, verbose = FALSE, rand.psoptim = NULL)
```

Arguments

<code>formula</code>	a symbolic description for the model to be estimated. Use $y \sim x \mid zu \mid zw$ to allow determinants zu of the first one-sided component and zw of the second.
<code>model_name</code>	model name for the estimation: "TTNE" for the Normal-Exponential-Exponential two-tier model, "TTHN" for the Normal-Half Normal-Half Normal two-tier model, or "TTNLS" for a nonlinear-least-squares two-tier estimator with no distributional assumption on u/w beyond their means. Matching ignores case, so "nhn", "NHN" and "Nhn" are the same choice.
<code>data</code>	A data frame containing the variables named in <code>formula</code> .
<code>z_link</code>	Scale on which the variance-determinant linear predictors are placed. "sd" (the default, and <code>ttsfm</code> 's historical convention) sets $\sigma_u = \exp(z'_u \delta)$ and likewise for σ_w ; "var" sets $\sigma_u = \sqrt{\exp(z'_u \delta)}$, matching <code>psfm</code> and the competing packages. The two are reparameterizations of the same model – identical maximised log-likelihood and identical frontier coefficients – but δ under the SD link is exactly half δ under the variance link, so a δ from <code>ttsfm</code> is not comparable with one from <code>psfm</code> or sfaR unless the same convention is used. See <code>sfm</code> , which gained the same argument.
<code>maxit.bobyqa</code>	Maximum number of iterations for the bobyqa optimization routine
<code>maxit.psoptim</code>	Maximum number of iterations for the psoptim optimization routine
<code>maxit.optim</code>	Maximum number of iterations for the optim optimization routine
<code>REPORT</code>	reporting parameter
<code>trace</code>	Integer. Tracing level passed through to the optimizer; larger values print more.
<code>pgtol</code>	Numeric. Projected-gradient tolerance passed to <code>optim()</code> 's "L-BFGS-B" method. 0 uses the optimizer's own default.
<code>start_val</code>	starting value (optional)
<code>PSopt</code>	use psoptim optimization routine (T or F)
<code>optHessian</code>	Logical. Should a numerically differentiated Hessian matrix be returned while using the optim routine? (for optim routine)
<code>inefdec</code>	Production or cost function
<code>upper</code>	Vector of upper values for the optim package.
<code>Method</code>	The method to be used for optim. See 'Details' within optim.
<code>logit</code>	Choice of using logit function
<code>verbose</code>	Logical. Print optimization progress messages? Default is FALSE.
<code>rand.psoptim</code>	Integer. seed for replication of psoptim. Default to NULL.

Details

The two-tier stochastic frontier model (Polachek and Yoon, 1987) models the composed error as $v+u-w$, where v is a symmetric normal noise term and u, w are independent one-sided components. "TTNE" assumes u and w are exponential; "TTHN" assumes they are half-normal; "TTNLS" makes no distributional assumption and instead estimates σ_u/σ_w by nonlinear least squares via the scaling property. Determinants of u and w can be included via the second and third parts of a two-pipe formula, which parameterize σ_u and σ_w through a log link.

Like the other composed-error models in this package, all three two-tier likelihoods can have a genuine (non-error) boundary mode where a variance component is estimated near its numerical floor in some samples; spot-check fits before serious use. "TTHN"'s post-estimation information-deficiency metrics (the `metrics` component of the returned object) involve a large number of bivariate-normal-CDF evaluations and are more experimental than the parameter estimates themselves – NULL if the underlying calculation fails. For "TTNLS", the reported standard error for the σ_v -position parameter (present only for layout consistency with "TTNE"/"TTHN", unused in the NLS objective) is always NA by construction.

Note on default iteration counts: like `sfm()/zsfm()`, `ttsfm()`'s likelihoods are closed-form, so its optimization defaults are set much higher than `psfm()`'s (which uses simulated maximum likelihood for several models and defaults to far fewer iterations for speed). See [psfm's](#) Details for more.

Value

An object of class "sfareg" containing the following components:

<code>out</code>	A matrix with parameter estimates, standard errors, and t-values.
<code>opt</code>	A list containing the optimization results from the final optimization procedure.
<code>total_time</code>	The total computation time for model estimation.
<code>start_v</code>	The starting values used in the optimization.
<code>model_name</code>	The name of the two-tier stochastic frontier model estimated (TTNE, TTHN, or TTNLS).
<code>formula</code>	The formula used in the model specification.
<code>coefficients</code>	A vector of estimated parameters.
<code>std.errors</code>	A vector of standard errors for the estimated parameters (NA if <code>optHessian = FALSE</code>).
<code>t.values</code>	A vector of t-values for the estimated parameters (NA if <code>optHessian = FALSE</code>).
<code>metrics</code>	A list of information-deficiency measures (M1, M2, M5, M6, M7, M10; see Polachek and Yoon) evaluated at the final parameter estimates. NULL if the underlying calculation failed (see Details for the "TTHN" caveat).
<code>call</code>	The matched call.

Note

Standard errors require `optHessian` set to TRUE

Author(s)

David H. Bernstein and Christopher F. Parmeter

References

Polachek, S. and Yoon, B.J. (1987). A Two-Tiered Earnings Frontier Estimation of Employer and Employee Information in the Labor Market. *The Review of Economics and Statistics*.

Examples

```
library(sfa)

data <- data_gen_cs(N = 200, rand = 1, sig_u = 0.3, sig_v = 0.2,
                    cons = 2, beta1 = 0.5, beta2 = 0.3, a = 5, mu = 0)

eq <- y_pcs ~ x1 + x2

ttsfm(formula      = eq,
      model_name = "TTNE",
      data        = data)
```

uhet_test	<i>Test whether technical inefficiency depends on firm characteristics</i>
-----------	--

Description

The two-step test of Kim and Schmidt (2008): regress the JLMS predictor on firm characteristics and test the slopes, with the variance correction that makes the test valid.

Usage

```
uhet_test(object, z, data = NULL)
```

Arguments

object	An "sfareg" fit from sfm with model_name = "NHN" or "NE", fitted <i>without</i> uhet and with keep_objective = TRUE. This is the first step, estimated under the null.
z	A one-sided formula (e.g. ~ z1 + z2) or a numeric matrix of the characteristics whose effect on inefficiency is being tested.
data	The data the model was fitted to.

Details

The procedure applied research actually runs is: fit the frontier, take $\hat{u}_i = E[u_i \mid \varepsilon_i]$, regress it on z_i , and test the slopes. **As usually performed that test is invalid.** $\hat{u}$ is a *generated* dependent variable, so the first-step estimation error in ψ enters the second-step variance; the naive test omits it. Kim and Schmidt show the omitted term is Gr_i with $G = E[z_i \nabla_\psi b_i]$, so the naive test is valid only when $G = 0$ – which holds when z is independent of x , and generally fails when they are correlated, the case applied work is usually in. There is no distribution-free correction, which is why only the two cases the paper works through are implemented.

What the correction buys. With z independent of x the two tests agree, as the theory says they must. With z strongly correlated with x ($\rho = 0.9$) the naive standard error is **52% too large** at $n = 800$ and the naive test's size collapses to **0.000** at a nominal 5%, while the corrected standard

error matches the true sampling spread to within half a percent (ratio 0.995) and the test retains size 0.032. A test that never rejects also has no power against local alternatives.

An honest caveat on size. The corrected test is *conservative* in these simulations: on Kim and Schmidt’s own base case (scaled exponential, $\sigma_v^2 = \lambda = 1$, $\rho = 0.5$, 1000 replications) it rejects 0.024, 0.036 and 0.028 of the time at $N = 200, 500, 1000$ against a nominal 0.05, where the paper reports 0.0495 for its corrected two-step test at $N = 200$. The naive branch reproduces the paper’s figure for the *uncorrected* test closely (0.023–0.029 against their 0.0237). The implementation was checked line by line against equations (5)–(13), the first-step scores agree with an analytic half-normal score to 10^{-11} , $\hat{\gamma}$ is unbiased and Gaussian (kurtosis 2.92), and an oracle test using the true sampling standard deviation has correct size (0.045) – so the discrepancy is in the variance estimate, not in $\hat{\gamma}$ or the reference distribution: $\text{cor}(|\hat{\gamma}|, \text{se})$ is 0.375, which compresses the ratio to a standard deviation of 0.93.

Two candidate explanations were tested and **both refuted**. Replacing the robust second-step variance with the homoskedastic one – the difference between the paper’s “OLS” and “OLS-H” variants – changes nothing (0.023/0.027/0.036 against 0.022/0.036/0.031). Nor does using the Hessian information in place of the outer-product one when forming r_i (0.020/0.026/0.026). The discrepancy is therefore **unexplained**, not merely unverified. **Treat a rejection as meaningful and a non-rejection as weak evidence.**

Fits sitting on the wrong-skew boundary are refused rather than answered: $\hat{u}$ is then identical for every firm, so there is no estimated inefficiency for z to explain, and the information matrix is singular there (Waldman 1982).

Value

A data frame with the corrected and naive statistics, their common degrees of freedom and p-values. The second-step coefficients and both sets of standard errors are attached as attributes.

References

- Kim, M. and Schmidt, P. (2008). Valid tests of whether technical inefficiency depends on firm characteristics. *Journal of Econometrics* **144**, 409–427.
- Wooldridge, J. M. (2002). *Econometric Analysis of Cross Section and Panel Data*, pp. 139–141.

See Also

[sfm](#) (its `uhet` argument fits the alternative directly), [marginal_effects](#), [skewness_test](#)

Examples

```
set.seed(2)
n <- 400
x1 <- rnorm(n); x2 <- rnorm(n)
z <- 0.9 * x1 + sqrt(1 - 0.81) * rnorm(n) # correlated with x: G != 0
y <- 0.5 + 0.5 * x1 + 0.5 * x2 + rnorm(n, 0, 0.5) - abs(rnorm(n, 0, 1))
d <- data.frame(y, x1, x2, z)

f <- sfm(y ~ x1 + x2, model_name = "NHN", data = d, keep_objective = TRUE)
uhet_test(f, ~z, data = d)
```

USUtilities

*USUtilities***Description**

Panel data on U.S. investor-owned fossil fuel-fired steam electric utilities for the period 1986-1997. These data include measures of output, capital, labour and maintenance, and fuel.

Usage

```
data("USUtilities")
```

Format

A data frame with 972 observations (a balanced panel of observations on 81 utilities over 12 years) on the following 7 variables.

firmID a numeric vector containing a unique firm identifier

year a numeric vector containing the year of the observation

q a numeric vector containing net steam electric power generation (MWh)

K a numeric vector containing capital stock, calculated using a method described by Christensen and Jorgenson (1970)

L a numeric vector containing quantity of labor and maintenance, calculated as cost divided by price index

F a numeric vector containing quantity of fuel used, calculated as fuel costs divided by fuel price index

trend a numeric vector containing an annual time trend (1992=100)

Details

The dataset covers 81 investor-owned utilities after aggregating subsidiaries and excluding plants in states with partial deregulation plans. Data sources include the Energy Information Administration (EIA), Federal Energy Regulatory Commission (FERC), and Bureau of Labor Statistics (BLS). Output is net steam electric generation from fossil fuel-fired boilers.

Source

Rungsuriyawiboon, S. and Stefanou, S.E. (2007). 'Dynamic Efficiency Estimation: An Application to U.S. Electric Utilities.' *Journal of Business & Economic Statistics*, 25(2), pp. 226-238. doi:10.1198/073500106000000288

References

Christensen, L.R. and Jorgenson, D.W. (1970). 'U.S. Real Product and Real Factor Input, 1928-1967.' *Review of Income and Wealth*, 16(1), pp. 19-50. doi: 10.1111/j.1475-4991.1970.tb00695.x

Examples

```
data(USUtilities)
```

zsfm

Zero-Inefficiency Stochastic Frontier Models

Description

Fits the zero-inefficiency stochastic frontier model of Kumbhakar, Parmeter and Tsionas (2013), in which the sample is a mixture of fully efficient firms and firms drawn from an ordinary stochastic frontier.

Usage

```
zsfm(formula, model_name = c("ZISF", "ZISF_Z"),
data, maxit.bobyqa = 10000, maxit.psoptim = 1000, maxit.optim = 1000,
REPORT = 1, trace = 0, pgtol = 0, start_val = FALSE, PSopt = FALSE,
optHessian = TRUE, inefdec = TRUE, upper = NA,
Method = "L-BFGS-B", logit = TRUE, verbose=FALSE, rand.psoptim = NULL)
```

Arguments

formula	a symbolic description for the model to be estimated
model_name	Which mixture specification to fit. "ZISF" holds the probability of being fully efficient constant across the sample; "ZISF_Z" lets that probability depend on covariates supplied in the second part of the formula ($y \sim x \mid z$). See 'Details'. Matching ignores case, so "nhn", "NHN" and "Nhn" are the same choice.
data	A data frame containing the variables named in formula.
maxit.bobyqa	Maximum number of iterations for the bobyqa optimization routine
maxit.psoptim	Maximum number of iterations for the psoptim optimization routine
maxit.optim	Maximum number of iterations for the optim optimization routine
REPORT	reporting parameter
trace	Integer. Tracing level passed through to the optimizer; larger values print more. 0 (the default) is silent.
pgtol	Numeric. Projected-gradient tolerance passed to optim()'s "L-BFGS-B" method. 0 uses the optimizer's own default.
start_val	starting value (optional)
PSopt	use psoptim optimization routine (T or F)
optHessian	Logical. Should a numerically differentiated Hessian matrix be returned while using the optim routine? (for optim routine)
inefdec	Production or cost function
upper	Vector of upper values for the optim package.

Method	The method to be used for optim. See 'Details' within optim.
logit	Logical, "ZISF_Z" only. Link mapping the linear predictor $z'\gamma$ to the probability of full efficiency. TRUE (the default) uses the logistic link $\Lambda(z'\gamma)$, which covers $(0, 1)$. FALSE uses $\Phi(z'\gamma)/(1 + \Phi(z'\gamma))$, which is bounded above by 0.5 : under that link no observation can be assigned a probability of full efficiency exceeding one half, whatever the data. This is not the probit analogue of the default and is retained only for backward compatibility; prefer logit = TRUE unless you specifically want that restriction.
verbose	Logical. Print optimization progress messages? Default is FALSE.
rand.psoptim	Integer. seed for replication of psoptim. Default to NULL.

Details

The zero-inefficiency stochastic frontier model treats the sample as a mixture of two regimes: with probability p a firm sits exactly on the frontier ($u = 0$, noise only), and with probability $1 - p$ it is drawn from an ordinary normal-half normal frontier. It is the natural specification when some fraction of the sample is thought to be fully efficient, which an ordinary frontier cannot represent because it forces $u > 0$ almost surely.

Under "ZISF" the mixing probability is a single constant, parameterized as $p = \exp(-|\gamma|)$. That parameterization is exactly symmetric in γ : $+\gamma$ and $-\gamma$ give identical fits and the optimizer may return either, so the sign of the reported gamma carries no meaning. Under "ZISF_Z" the probability varies with covariates z through the link chosen by logit.

post.prob reports the posterior probability that each observation belongs to the fully efficient regime, given its residual; jlms reports the Jondrow et al. (1982) inefficiency prediction.

Note on default iteration counts: like sfm(), zsfm()'s likelihood is closed-form, so its optimization defaults (thousands of iterations) are set much higher than psfm()'s (which uses simulated maximum likelihood for several models and defaults to far fewer iterations for speed). See psfm's Details for more.

Value

An object of class "sfareg" containing the following components:

out	A matrix with parameter estimates, standard errors, and t-values.
opt	A list containing the optimization results from the final optimization procedure.
total_time	The total computation time for model estimation.
start_v	The starting values used in the optimization.
model_name	The name of the zero-inefficiency stochastic frontier model estimated ("ZISF" or "ZISF_Z").
formula	The formula used in the model specification.
jlms	Predicted technical efficiency using the Jondrow et al. (1982) conditional mean estimator (JLMS).
post.prob	Posterior probabilities of being fully efficient.
coefficients	A vector of estimated parameters.

<code>std.errors</code>	A vector of standard errors for the estimated parameters (NA if <code>optHessian = FALSE</code>).
<code>t.values</code>	A vector of t-values for the estimated parameters (NA if <code>optHessian = FALSE</code>).
<code>call</code>	The matched call.

Note

Standard errors require `optHessian` set to `TRUE`

Author(s)

Christopher F. Parmeter and David H. Bernstein

References

Kumbhakar, S.C., Parmeter, C.F. and Tsionas, E.G. (2013) 'A zero inefficiency stochastic frontier model', *Journal of Econometrics*, 172(1), pp. 66-76. doi:10.1016/j.jeconom.2012.08.021.

See Also

[sfm](#), [psfm](#), [ttsfm](#), [sfareg-methods](#), [panel89](#)

Examples

```
library(sfa)

eqz    <- y ~ q1 + q2 + q3 + q4 + q5 + w1 + w2 + w3 + w4 | z

data(panel89)

zsfm(formula    = eqz,
      model_name = "ZISF_Z",
      data       = panel89,
      logit      = TRUE)
```

Index

- * **datasets**
 - FinnishElec, [27](#)
 - Indian, [33](#)
 - panel89, [57](#)
 - USUtilities, [120](#)
 - * **distribution**
 - pcomposed, [58](#)
 - * **htest**
 - endogeneity_test, [23](#)
 - lcsfm_homogeneity, [44](#)
 - psfm_bootstrap, [80](#)
 - spec_test, [110](#)
 - * **models**
 - copsfm, [7](#)
 - efficiency, [20](#)
 - efficiency_ci, [22](#)
 - ivsfm, [37](#)
 - marginal_effects, [47](#)
 - meanefficiency, [48](#)
 - moment_range, [50](#)
 - psfm_bootstrap, [80](#)
 - selsfm, [85](#)
 - sfma, [104](#)
 - simulation_se, [106](#)
 - TIC, [113](#)
 - * **package**
 - sfa-package, [3](#)
 - * **regression**
 - influence_sfa, [35](#)
- AIC, [113–115](#)
- bread.sfareg (sfareg-methods), [89](#)
- calibrate_c, [5](#), [20](#), [31](#), [32](#), [36](#)
- clusterCall, [82](#)
- coef.sfareg (sfareg-methods), [89](#)
- composed_cdf (pcomposed_model), [60](#)
- copsfm, [7](#), [59](#)
- cw_test, [10](#)
- data_gen_cs, [12](#), [19](#), [56](#), [104](#)
- data_gen_p, [16](#), [16](#), [62](#), [79](#)
- dcomposed, [61](#)
- dcomposed (pcomposed), [58](#)
- density_weights, [7](#), [19](#), [31](#), [32](#), [36](#), [37](#)
- efficiency, [20](#), [49](#)
- efficiency_ci, [21](#), [22](#), [48](#)
- endogeneity_test, [23](#), [39](#), [40](#)
- ercomp, [79](#)
- esfm, [25](#), [51](#)
- estfun, [36](#)
- estfun.sfareg (sfareg-methods), [89](#)
- FinnishElec, [27](#)
- fitted.sfareg (predict.sfareg), [63](#)
- Formula, [84](#)
- gof_test, [11](#), [12](#), [28](#)
- hscore, [30](#), [31](#), [32](#)
- hscore_select, [6](#), [7](#), [20](#), [31](#), [31](#), [36](#), [37](#)
- Indian, [33](#)
- inefficiency_test, [24](#), [26](#), [34](#)
- influence_sfa, [35](#)
- ivsfm, [23](#), [24](#), [37](#)
- jacobian, [101](#)
- lcsfm, [40](#), [44–46](#)
- lcsfm_homogeneity, [41](#), [42](#), [44](#), [44](#)
- logLik, [39](#)
- logLik.sfareg (sfareg-methods), [89](#)
- makeCluster, [81](#)
- marginal_effects, [47](#), [97](#), [99](#), [119](#)
- meanefficiency, [48](#)
- moment_range, [24](#), [50](#), [109](#), [112](#)
- nobs.sfareg (sfareg-methods), [89](#)

- npsfm, 52
- panel89, 57, 123
- parLapply, 83, 84
- pblapply, 83, 84
- pcomposed, 58, 60, 61
- pcomposed_model, 29, 60
- pdata.frame, 17, 19
- PL80_MVTN, 61, 66
- plm, 79
- plot.sfareg (sfa_diagnostics), 91
- pnorm, 59
- predict.sfareg, 63
- print.cw_test (cw_test), 10
- print.esfm (esfm), 25
- print.npsfareg (npsfm), 52
- print.sfa_endog_test
 (endogeneity_test), 23
- print.sfa_hscore (hscore_select), 31
- print.sfa_influence (influence_sfa), 35
- print.sfa_mlrt (lcsfm_homogeneity), 44
- print.sfa_moment_range (moment_range),
 50
- print.sfa_spec_grid (spec_test), 110
- print.sfa_spec_test (spec_test), 110
- print.sfa_vuong (TIC), 113
- print.sfdiag (sfa_diagnostics), 91
- print.sfareg, 64, 90, 113
- print.sfma (sfma), 104
- psfm, 18, 19, 38, 44, 47, 48, 56, 62–64, 65, 81,
 82, 84, 89, 90, 97, 99, 101, 104, 112,
 114, 116, 117, 122, 123
- psfm_bootstrap, 79, 80
- rcauchy, 16
- residuals.sfareg (predict.sfareg), 63
- rexp, 16
- rnorm, 16
- rt, 16
- rtruncnorm, 16, 83
- runif, 16
- selsfm, 85
- sfa, 81
- sfa-package, 3
- sfa_diagnostics, 23, 68, 91, 96, 107, 109,
 115
- sfareg-methods, 89
- sfm, 8–10, 14–16, 22, 23, 25, 26, 28, 29, 31,
 32, 34–38, 40, 44, 47, 48, 54, 56,
 59–61, 63, 64, 67, 79, 88–90, 92, 93,
 105, 106, 108, 109, 112, 114–116,
 118, 119, 123
- sfma, 104
- simulation_se, 106
- skewness_test, 12, 25, 26, 34, 35, 46, 51, 73,
 74, 108, 111, 112, 115, 119
- spec_test, 12, 24, 29, 35, 51, 106, 109, 110
- spec_test_all (spec_test), 110
- summary.npsfareg (npsfm), 52
- summary.sfareg, 64, 65, 90, 92, 112
- symmetry_test (esfm), 25
- TIC, 46, 104, 106, 112, 113
- ttsfm, 13, 15, 16, 44, 63, 64, 89, 90, 99, 104,
 112, 115, 123
- uhet_test, 118
- USUtilities, 120
- vcov, 36
- vcov.sfareg (sfareg-methods), 89
- vuong, 104, 106
- vuong (TIC), 113
- warning, 83
- zsfm, 15, 16, 42, 44, 63, 64, 89, 90, 104, 112,
 121