Package: prodest (via r-universe)

August 28, 2024

Type Package

Title Production Function Estimation

Version 0.1.1

Author Gabriele Rovigatti [aut,cre]

Maintainer Gabriele Rovigatti <gabriele.rovigatti@gmail.com>

Description TFP estimation with the control function approach.

License GPL-3

BugReports https://github.com/

URL https://github.com/

LazyData TRUE

Depends R (>= 2.10), Rsolnp, DEoptim, dplyr, parallel, Matrix, methods

Suggests testthat

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

RemoteUrl https://github.com/gabrielerovigatti/prodest

RemoteRef HEAD

RemoteSha 9e5364c5a011425abe8adc67bead088616a8467c

Contents

chilean *Data: Chilean firm-level production data 1986-1996*

Description

Sectoral subsample of Chilean firm-level production data 1986-1996.

Usage

```
data("chilean")
```

Format

A data.frame object containing 9 variables with production-related data.

Value

`Y`	vector of log(outcome) - Value added.
`sX`	vector of log(capital).
`fX`	matrix of log(skilled labor) and log(unskilled labor).
`cX`	vector of log(water).
`pX`	vector of log(electricity).
`inv`	vector of log(investment).
`idvar`	vector of panel identifier.
`timevar`	vector of time.

References

http://www.ine.cl/canales/chile_estadistico/estadisticas_economicas/industria/series_estadisticas/series_estadisticas_enia.php

panelSim *Simulate Panel dataset*

Description

`panelSim()` produces a N*T balanced panel dataset of firms' production. In particular, it returns a `data.frame` with free, state and proxy variables aimed at performing Monte Carlo simulations on productivity-related models.

Usage

```
panelSim(N = 1000, T = 100, alphaL = .6, alphaK = .4, DGP = 1,
         rho = .7, sigeps = .1, sigomg = .3, rholnw = .3, seed = 123456)
```

Arguments

`N`	the number of firms. By default `N=1000`
`T`	the total time span to be simulated. Only a fraction (the last 10% of observations) will be returned. By default `T=100`
`alphaL`	the parameter of the free variable. By default `alphaL=.6`
`alphaK`	the parameter of the state variable. By default `alphaK=.4`
`DGP`	Type of DGP; accepts 1, 2 or 3. They differ in terms of shock to wages (0 or 0.1), Δ (0 or 0.5) and shock to labor (0 or 0.37). See $details$. By default `DGP=1`.
`rho`	the AR(1) coefficient for omega. By default `rho=0.7`
`sigeps`	the standard deviation of epsilon. See $details$. By default `sigeps = .1`.
`sigomg`	the standard deviation of the innovation to productivity ω. By default `sigomg = .3`.
`rholnw`	AR(1) coefficient for log(wage). By default `rholnw=.3`.
`seed`	seed set when the routine starts. By default `seed = 123456`.

Details

`panelSim()` is the R implementation of the DGP written by Ackerberg, Caves and Frazer (2015).

Value

$panelSim()$ returns a `data.frame` with 7 variables:

- $idvar$ ID codes from 1 to N (by default $N = 1000$).
- $timevar$ time variable ranging 1 to $round(T*0.1)$ (by default $T = 100$ and $max(timevar) = 10$).
- Y log output value added variable
- sX log state variable
- fX log free variable
- $pX1$ log proxy variable - no measurement error
- $pX2$ log proxy variable - $\sigma_{measurementerror} = .1$
- $pX3$ log proxy variable - $\sigma_{measurementerror} = .2$
- $pX4$ log proxy variable - $\sigma_{measurementerror} = .5$

Author(s)

Gabriele Rovigatti

References

Ackerberg, D., Caves, K. and Frazer, G. (2015). "Identi

cation properties of recent production function estimators." Econometrica, 83(6), 2411-2451.

Examples

```
require(prodest)

## Simulate a dataset with 1000 firms (T = 100). \code{Panelsim()} delivers the last 10% of usable time per panel.

panel.data <- panelSim()
attach(panel.data)

## Estimate various models
ACF.fit <- prodestACF(Y, fX, sX, pX2, idvar, timevar, theta0 = c(.5,.5))
LP.fit <- prodestLP(Y, fX, sX, pX2, idvar, timevar)
WRDG.fit <- prodestWRDG(Y, fX, sX, pX3, idvar, timevar, R = 5)

## print results in lateX tabular format
printProd(list(LP.fit, ACF.fit, WRDG.fit))
```

---

printProd — *Print to lateX - prod objects*

Description

The `printProd()` function accepts a `list` of `prod` class objects and returns a screen printed tabular in lateX format of the results.

Usage

```
printProd(mods, modnames = NULL, parnames = NULL, outfile = NULL, ptime = FALSE, nboot = FALSE)
```

Arguments

`mods`	a `list` of `prod` objects.
`modnames`	an optional vector of model names. By default, model names are the `@ModelMethod` values in `prod` objects.
`parnames`	an optional vector of parameter names. By default, parameter names are the `names()` vector of `@Estimatespars` in `prod` objects.
`outfile`	optional string with the path and directory to store a text file (.txt, .tex, etc. depending on the specified extension) with the tabular. By default `outfile = NULL`.
`ptime`	add a row showing the computational time. By default `ptime = FALSE`.
`nboot`	add a row showing the number of bootstrap repetitions. By default `nboot = FALSE`.

Value

The output of the function `printProd` is either a screen printed tabular in lateX format of `prod` object results or a text file tabular in lateX format of `prod` object results.

Author(s)

Gabriele Rovigatti

Examples

```
data("chilean")
WRDGfit <- prodestWRDG_GMM(d$Y, fX = cbind(d$fX1, d$fX2), d$sX, d$pX, d$idvar, d$timevar)
OPfit <- prodestOP(d$Y, fX = cbind(d$fX1, d$fX2), d$sX, d$pX, d$idvar, d$timevar)
printProd(list(OPfit, WRDGfit), modnames = c('Olley-Pakes', 'Wooldridge'), parnames = c('bunsk', 'bsk', 'bk'))
```

---

prod *Class for Prodest Fitted object*

Description

Class for prodest fitted objects.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`Model`**:** Object of class `list`. Contains information about the model and the optimization procedue:

- `method`: `string` The method used in estimation.
- `FSbetas`: `numeric` First-stage estimated parameters.
- `boot.repetitions`: `numeric` Number of bootstrap repetitions.
- `elapsed.time`: `numeric` Time - in seconds - required for estimation.
- `theta0`: `numeric` Vector of Second-stage optimization starting points.
- `opt`: `string` Optimizer used for the Second-stage.
- `seed`: `numeric` seed set.
- `opt.outcome`: `list` Optimization outcome (depends on optimizer choice).

`Data`**:** Object of class `list`. Contains:

- `Y`: `numeric` Dependent variable - Value added.
- `free`: `matrix` Free variable(s).
- `state`: `matrix` State variable(s).
- `proxy`: `matrix` Proxy variable(s).
- `control`: `matrix` Control variable(s).
- `idvar`: `numeric` Panel identifiers.
- `timevar`: `numeric` Time identifiers.
- `FSresiduals`: `numeric` First-Stage residuals.

`Estimates`**:** Object of class `list`. Contains:

- `pars`: `numeric` Estimated parameters for the variables of interest.
- `std.errors`: `numeric` Estimated standard errors for the variables of interest.

Methods

- show signature(object = 'prod'): Show table with the method, the estimated parameters and their standard errors.
- summary signature(object = 'prod'): Show table with method, parameters, std.errors and auxiliary information on model and optimization.
- FSres signature(object = 'prod'): Extract First-Stage residual vector.
- omega signature(object = 'prod'): Extract estimated productivity vector.
- coef signature(object = 'prod'): Extract estimated coefficients.

Author(s)

Gabriele Rovigatti

---

prodestACF *Estimate productivity - Ackerberg-Caves-Frazer correction*

Description

The prodestACF() function accepts at least 6 objects (id, time, output, free, state and proxy variables), and returns a prod object of class S3 with three elements: (i) a list of model-related objects, (ii) a list with the data used in the estimation and estimated vectors of first-stage residuals, and (iii) a list with the estimated parameters and their bootstrapped standard errors .

Usage

```
prodestACF(Y, fX, sX, pX, idvar, timevar, zX = NULL, control = c('none','fs','2s'), dum = F, G = 3, A = 3
                      theta0 = NULL, seed = 123456, cluster = NULL)
```

Arguments

Y	the vector of value added log output.
fX	the vector/matrix/dataframe of log free variables.
sX	the vector/matrix/dataframe of log state variables.
pX	the vector/matrix/dataframe of log proxy variables.
idvar	the vector/matrix/dataframe identifying individual panels.
timevar	the vector/matrix/dataframe identifying time.
zX	the vector/matrix/dataframe of (input price) control variables.
control	the way in which the control variables should be included. By default control = 'none'. Specifying control = 'fs' only includes controls in the first-stage polynomial, as in De Loecker and Warzynski (2012). Specifying control = 'fs' includes controls in both first and second stage, as in De Loecker, Goldberg, Khandelwal and Pavcnik (2016). Note that this is not desirable when estimating a revenue gross ouput production function, as in that case not controlling for input prices has the coincidental benefit that the input price bias partially cancels out the output price bias.

`dum`	whether time dummies should be included in the first stage. By default `dum = F`.
`G`	the degree of the first-stage polynomial in fX, sX and pX. By default `G = 3`.
`A`	the degree of the polynomial for the Markov productivity process. By default `A = 3`.
`R`	the number of block bootstrap repetitions to be performed in the standard error estimation. By default `R = 20`.
`orth`	a Boolean that determines whether first-stage polynomial should be orthogonal or raw. By default, `orth = F`. It is recommended to set orth to T if degree of polynomial is high.
`opt`	a string with the optimization algorithm to be used during the estimation. By default `opt = 'optim'`.
`theta0`	a vector with the second stage optimization starting points. By default `theta0 = NULL` and the optimization is run starting from the first stage estimated parameters + $N(0, 0.01)$ noise.
`cluster`	an object of class `"SOCKcluster"` or `"cluster"`. By default `cluster = NULL`.
`seed`	seed set when the routine starts. By default `seed = 123456`.

Details

Consider a Cobb-Douglas production technology for firm i at time t

- $y_{it} = \alpha + w_{it}\beta + k_{it}\gamma + \omega_{it} + \epsilon_{it}$

where y_{it} is the (log) output, w_it a 1xJ vector of (log) free variables, k_it is a 1xK vector of state variables and ϵ_{it} is a normally distributed idiosyncratic error term. The unobserved technical efficiency parameter ω_{it} evolves according to a first-order Markov process:

- $\omega_{it} = E(\omega_{it}|\omega_{it-1}) + u_{it} = g(\omega_{it-1}) + u_{it}$

and u_{it} is a random shock component assumed to be uncorrelated with the technicalefficiency, the state variables in k_{it} and the lagged free variables w_{it-1}. ACF propose an estimation algorithm alternative to OP and LP procedures claiming that the labour demand and the control function are partially collinear. It is based on the following set of assumptions:

- a) $p_{it} = p(k_{it}, l_{it}, \omega_{it})$ is the proxy variable policy function;
- b) p_{it} is strictly monotone in ω_{it};
- c) ω_{it} is scalar unobservable in $p_{it} = m(.)$;
- d) The state variable are decided at time t-1. The less variable labor input, l_{it}, is chosen at t-b, where $0 < b < 1$. The free variables, w_{it}, are chosen in t when the firm productivity shock is realized.

Under this set of assumptions, the first stage is meant to remove the shock ϵ_{it} from the the output, y_{it}. As in the OP/LP case, the inverted policy function replaces the productivity term ω_{it} in the production function:

- $y_{it} = k_{it}\gamma + w_{it}\beta + l_{it}\mu + h(p_{it}, k_{it}, w_{it}, l_{it}) + \epsilon_{it}$

which is estimated by a non-parametric approach - First Stage. Exploiting the Markovian nature of the productivity process one can use assumption d) in order to set up the relevant moment conditions and estimate the production function parameters - Second stage.

Value

The output of the function prodestACF is a member of the S3 class **prod**. More precisely, is a list (of length 3) containing the following elements:

Model, a list with elements:

- method: a string describing the method ('ACF').
- boot.repetitions: the number of bootstrap repetitions used for standard errors' computation.
- elapsed.time: time elapsed during the estimation.
- theta0: numeric object with the optimization starting points - second stage.
- opt: string with the optimization routine used - 'optim', 'solnp' or 'DEoptim'.
- seed: the seed set at the beginning of the estimation.
- opt.outcome: optimization outcome.
- FSbetas: first stage estimated parameters.

Data, a list with elements:

- Y: the vector of value added log output.
- free: the vector/matrix/dataframe of log free variables.
- state: the vector/matrix/dataframe of log state variables.
- proxy: the vector/matrix/dataframe of log proxy variables.
- control: the vector/matrix/dataframe of log control variables.
- idvar: the vector/matrix/dataframe identifying individual panels.
- timevar: the vector/matrix/dataframe identifying time.
- FSresiduals: numeric object with the residuals of the first stage.

Estimates, a list with elements:

- pars: the vector of estimated coefficients.
- std.errors: the vector of bootstrapped standard errors.

Members of class prod have an omega method returning a numeric object with the estimated productivity - that is: $\omega_{it} = y_{it} - (\alpha + w_{it}\beta + k_{it}\gamma)$. FSres method returns a numeric object with the residuals of the first stage regression, while summary, show and coef methods are implemented and work as usual.

Author(s)

Gabriele Rovigatti

References

Ackerberg, D., Caves, K. and Frazer, G. (2015). "Identification properties of recent production function estimators." Econometrica, 83(6), 2411-2451. De Loecker, J., Goldberg, P. K., Khandelwal, A. K., & Pavcnik, N. (2016). "Prices, markups, and trade reform." Econometrica, 84(2), 445-510. De Loecker, J., & Warzynski, F. (2012). "Markups and firm-level export status." American Economic Review, 102(6), 2437-71.

Examples

```
require(prodest)

## Chilean data on production.The full version is Publicly available at
## http://www.ine.cl/canales/chile_estadistico/estadisticas_economicas/industria/series_estadisticas/series_

data(chilean)

# we fit a model with two free (skilled and unskilled), one state (capital) and one proxy variable (electricity)

ACF.fit <- prodestACF(d$Y, fX = cbind(d$fX1, d$fX2), d$sX, d$pX, d$idvar, d$timevar, theta0 = c(.5,.5,.5), seed =
ACF.fit.solnp <- prodestACF(d$Y, fX = cbind(d$fX1, d$fX2), d$sX, d$pX, d$idvar, d$timevar, theta0 = c(.5,.5,.5),

# run the same regression in parallel
nCores <- as.numeric(Sys.getenv("NUMBER_OF_PROCESSORS"))
cl <- makeCluster(getOption("cl.cores", nCores - 1))
ACF.fit.par <- prodestACF(d$Y, fX = cbind(d$fX1, d$fX2), d$sX, d$pX, d$idvar, d$timevar, theta0 = c(.5,.5,.5), c
stopCluster(cl)

# show results
coef(ACF.fit)
coef(ACF.fit.solnp)
coef(ACF.fit.par)

 # show results in .tex tabular format
 printProd(list(ACF.fit, ACF.fit.solnp, ACF.fit.par))
```

---

prodestLP *Estimate productivity - Levinsohn-Petrin method*

Description

The prodestLP() The prodestWRDG() function accepts at least 6 objects (id, time, output, free, state and proxy variables), and returns a prod object of class S3 with three elements: (i) a list of model-related objects, (ii) a list with the data used in the estimation and estimated vectors of first-stage residuals, and (iii) a list with the estimated parameters and their bootstrapped standard errors.

Usage

```
prodestLP(Y, fX, sX, pX, idvar, timevar, R = 20, G = 3, orth = F, cX = NULL,
      opt = 'optim', theta0 = NULL, seed = 123456, cluster = NULL, tol = 1e-100)
```

Arguments

Y	the vector of value added log output.
fX	the vector/matrix/dataframe of log free variables.

`sX`	the vector/matrix/dataframe of log state variables.
`pX`	the vector/matrix/dataframe of log proxy variables.
`cX`	the vector/matrix/dataframe of control variables. By default `cX= NULL`.
`idvar`	the vector/matrix/dataframe identifying individual panels.
`timevar`	the vector/matrix/dataframe identifying time.
`R`	the number of block bootstrap repetitions to be performed in the standard error estimation. By default `R = 20`.
`G`	the degree of the first-stage polynomial in fX, sX and pX. By default `G = 3`.
`orth`	a Boolean that determines whether first-stage polynomial should be orthogonal or raw. By default, `orth = F`. It is recommended to set orth to T if degree of polynomial is high.
`opt`	a string with the optimization algorithm to be used during the estimation. By default `opt = 'optim'`.
`theta0`	a vector with the second stage optimization starting points. By default `theta0 = NULL` and the optimization is run starting from the first stage estimated parameters + $N(\mu = 0, \sigma = 0.01)$ noise.
`cluster`	an object of class `"SOCKcluster"` or `"cluster"`. By default `cluster = NULL`.
`seed`	seed set when the routine starts. By default `seed = 123456`.
`tol`	optimizer tolerance. By default `tol = 1e-100`.

Details

Consider a Cobb-Douglas production technology for firm i at time t

- $y_{it} = \alpha + w_{it}\beta + k_{it}\gamma + \omega_{it} + \epsilon_{it}$

where y_{it} is the (log) output, w_it a 1xJ vector of (log) free variables, k_it is a 1xK vector of state variables and ϵ_{it} is a normally distributed idiosyncratic error term. The unobserved technical efficiency parameter ω_{it} evolves according to a first-order Markov process:

- $\omega_{it} = E(\omega_{it}|\omega_{it-1}) + u_{it} = g(\omega_{it-1}) + u_{it}$

and u_{it} is a random shock component assumed to be uncorrelated with the technicalefficiency, the state variables in k_{it} and the lagged free variables w_{it-1}. The LP method relies on the following set of assumptions:

- a) firms immediately adjust the level of inputs according to demand function $m(\omega_{it}, k_{it})$ after the technical efficiency shock realizes;
- b) m_{it} is strictly monotone in ω_{it};
- c) ω_{it} is scalar unobservable in $m_{it} = m(.)$;
- d) the levels of k_{it} are decided at time $t-1$; the level of the free variable, w_{it}, is decided after the shock u_{it} realizes.

Assumptions a)-d) ensure the invertibility of m_{it} in ω_{it} and lead to the partially identified model:

- $y_{it} = \alpha + w_{it}\beta + k_{it}\gamma + h(m_{it}, k_{it}) + \epsilon_{it} = \alpha + w_{it}\beta + \phi(m_{it}, k_{it}) + \epsilon_{it}$

which is estimated by a non-parametric approach - First Stage. Exploiting the Markovian nature of the productivity process one can use assumption d) in order to set up the relevant moment conditions and estimate the production function parameters - Second stage. Exploting the resisual ν_{it} of:

- $y_{it} - w_{it}\hat{\beta} = \alpha + k_{it}\gamma + g(\omega_{it-1}, \chi_{it}) + \nu_{it}$

and $g(.)$ is typically left unspecified and approximated by a n^{th} order polynomial and χ_{it} is an inidicator function for the attrition in the market.

Value

The output of the function `prodestLP` is a member of the `S3` class **prod**. More precisely, is a list (of length 3) containing the following elements:

`Model`, a list containing:

- `method`: a string describing the method ('LP').
- `boot.repetitions`: the number of bootstrap repetitions used for standard errors' computation.
- `elapsed.time`: time elapsed during the estimation.
- `theta0`: numeric object with the optimization starting points - second stage.
- `opt`: string with the optimization routine used - 'optim', 'solnp' or 'DEoptim'.
- `seed`: the seed set at the beginning of the estimation.
- `opt.outcome`: optimization outcome.
- `FSbetas`: first stage estimated parameters.

`Data`, a list containing:

- `Y`: the vector of value added log output.
- `free`: the vector/matrix/dataframe of log free variables.
- `state`: the vector/matrix/dataframe of log state variables.
- `proxy`: the vector/matrix/dataframe of log proxy variables.
- `control`: the vector/matrix/dataframe of log control variables.
- `idvar`: the vector/matrix/dataframe identifying individual panels.
- `timevar`: the vector/matrix/dataframe identifying time.
- `FSresiduals`: numeric object with the residuals of the first stage.

`Estimates`, a list containing:

- `pars`: the vector of estimated coefficients.
- `std.errors`: the vector of bootstrapped standard errors.

Members of class `prod` have an `omega` method returning a numeric object with the estimated productivity - that is: $\omega_{it} = y_{it} - (\alpha + w_{it}\beta + k_{it}\gamma)$. `FSres` method returns a numeric object with the residuals of the first stage regression, while `summary`, `show` and `coef` methods are implemented and work as usual.

Author(s)

Gabriele Rovigatti

References

Levinsohn, J. and Petrin, A. (2003). "Estimating production functions using inputs to control for unobservables." The Review of Economic Studies, 70(2), 317-341.

Examples

```
require(prodest)

## Chilean data on production.
## Publicly available at http://www.ine.cl/canales/chile_estadistico/estadisticas_economicas/industria/series

data(chilean)

## we fit a model with two free (skilled and unskilled), one state (capital) and one proxy variable (electricity)

LP.fit <- prodestLP(d$Y, fX = cbind(d$fX1, d$fX2), d$sX, d$pX, d$idvar, d$timevar, seed = 154673)
LP.fit.solnp <- prodestLP(d$Y, fX = cbind(d$fX1, d$fX2), d$sX, d$pX, d$idvar, d$timevar, opt = 'solnp')

# run the same model in parallel
require(parallel)
nCores <- as.numeric(Sys.getenv("NUMBER_OF_PROCESSORS"))
cl <- makeCluster(getOption("cl.cores", nCores - 1))
LP.fit.par <- prodestLP(d$Y, fX = cbind(d$fX1, d$fX2), d$sX, d$pX, d$idvar, d$timevar, cluster = cl, seed = 15467
stopCluster(cl)

# show results
summary(LP.fit)
summary(LP.fit.solnp)
summary(LP.fit.par)

# show results in .tex tabular format
 printProd(list(LP.fit, LP.fit.solnp, LP.fit.par))
```

---

prodestOP *Estimate productivity - Olley-Pakes method*

Description

The `prodestOP()` function accepts at least 6 objects (id, time, output, free, state and proxy variables), and returns a `prod` object of class `S4` with three elements: (i) a list of model-related objects, (ii) a list with the data used in the estimation and estimated vectors of first-stage residuals, and (iii) a list with the estimated parameters and their bootstrapped standard errors .

Usage

```
prodestOP(Y, fX, sX, pX, idvar, timevar, R = 20, G = 3, orth = F, cX = NULL,
      opt = 'optim', theta0 = NULL, seed = 123456, cluster = NULL, tol = 1e-100)
```

Arguments

`Y`	the vector of value added log output.
`fX`	the vector/matrix/dataframe of log free variables.
`sX`	the vector/matrix/dataframe of log state variables.
`pX`	the vector/matrix/dataframe of log proxy variables.
`cX`	the vector/matrix/dataframe of control variables. By default `cX= NULL`.
`idvar`	the vector/matrix/dataframe identifying individual panels.
`timevar`	the vector/matrix/dataframe identifying time.
`R`	the number of block bootstrap repetitions to be performed in the standard error estimation. By default `R = 20`.
`G`	the degree of the first-stage polynomial in fX, sX and pX. By default, `G = 3`.
`orth`	a Boolean that determines whether first-stage polynomial should be orthogonal or raw. By default, `orth = F`. It is recommended to set orth to T if degree of polynomial is high.
`opt`	a string with the optimization algorithm to be used during the estimation. By default `opt = 'optim'`.
`theta0`	a vector with the second stage optimization starting points. By default `theta0 = NULL` and the optimization is run starting from the first stage estimated parameters + $N(\mu = 0, \sigma = 0.01)$ noise.
`cluster`	an object of class `"SOCKcluster"` or `"cluster"`. By default `cluster = NULL`.
`seed`	seed set when the routine starts. By default `seed = 123456`.
`tol`	optimizer tolerance. By default `tol = 1e-100`.

Details

Consider a Cobb-Douglas production technology for firm i at time t

- $y_{it} = \alpha + w_{it}\beta + k_{it}\gamma + \omega_{it} + \epsilon_{it}$

where y_{it} is the (log) output, w_it a 1xJ vector of (log) free variables, k_it is a 1xK vector of state variables and ϵ_{it} is a normally distributed idiosyncratic error term. The unobserved technical efficiency parameter ω_{it} evolves according to a first-order Markov process:

- $\omega_{it} = E(\omega_{it}|\omega_{it-1}) + u_{it} = g(\omega_{it-1}) + u_{it}$

and u_{it} is a random shock component assumed to be uncorrelated with the technicalefficiency, the state variables in k_{it} and the lagged free variables w_{it-1}. The OP method relies on the following set of assumptions:

- a) $i_{it} = i(k_{it}, \omega_{it})$ - investments are a function of both the state variable and the technical efficiency parameter;

- b) i_{it} is strictly monotone in ω_{it};
- c) ω_{it} is scalar unobservable in $i_{it} = i(.)$;
- d) the levels of i_{it} and k_{it} are decided at time $t-1$; the level of the free variable, w_{it}, is decided after the shock u_{it} realizes.

Assumptions a)-d) ensure the invertibility of i_{it} in ω_{it} and lead to the partially identified model:

- $y_{it} = \alpha + w_{it}\beta + k_{it}\gamma + h(i_{it}, k_{it}) + \epsilon_{it} = \alpha + w_{it}\beta + \phi(i_{it}, k_{it}) + \epsilon_{it}$

which is estimated by a non-parametric approach - First Stage. Exploiting the Markovian nature of the productivity process one can use assumption d) in order to set up the relevant moment conditions and estimate the production function parameters - Second stage. Exploting the resisual e_{it} of:

- $y_{it} - w_{it}\hat{\beta} = \alpha + k_{it}\gamma + g(\omega_{it-1}, \chi_{it}) + \epsilon_{it}$

and $g(.)$ is typically left unspecified and approximated by a n^{th} order polynomial and χ_{it} is an inidicator function for the attrition in the market.

Value

The output of the function `prodestOP` is a member of the `S3` class **prod**. More precisely, is a list (of length 3) containing the following elements:

`Model`, a list containing:

- `method`: a string describing the method ('OP').
- `boot.repetitions`: the number of bootstrap repetitions used for standard errors' computation.
- `elapsed.time`: time elapsed during the estimation.
- `theta0`: numeric object with the optimization starting points - second stage.
- `opt`: string with the optimization routine used - 'optim', 'solnp' or 'DEoptim'.
- `seed`: the seed set at the beginning of the estimation.
- `opt.outcome`: optimization outcome.
- `FSbetas`: first stage estimated parameters.

`Data`, a list containing:

- `Y`: the vector of value added log output.
- `free`: the vector/matrix/dataframe of log free variables.
- `state`: the vector/matrix/dataframe of log state variables.
- `proxy`: the vector/matrix/dataframe of log proxy variables.
- `control`: the vector/matrix/dataframe of log control variables.
- `idvar`: the vector/matrix/dataframe identifying individual panels.
- `timevar`: the vector/matrix/dataframe identifying time.
- `FSresiduals`: numeric object with the residuals of the first stage.

`Estimates`, a list containing:

- `pars`: the vector of estimated coefficients.
- `std.errors`: the vector of bootstrapped standard errors.

Members of class `prod` have an `omega` method returning a numeric object with the estimated productivity - that is: $\omega_{it} = y_{it} - (\alpha + w_{it}\beta + k_{it}\gamma)$. `FSres` method returns a numeric object with the residuals of the first stage regression, while `summary`, `show` and `coef` methods are implemented and work as usual.

Author(s)

Gabriele Rovigatti

References

Olley, S G and Pakes, A (1996). "The dynamics of productivity in the telecommunications equipment industry." Econometrica, 64(6), 1263-1297.

Examples

```
  require(prodest)

  ## Chilean data on production.The full version is Publicly available at
## http://www.ine.cl/canales/chile_estadistico/estadisticas_economicas/industria/series_estadisticas/series_

  data(chilean)

# we fit a model with two free (skilled and unskilled), one state (capital) and one proxy variable (electricity)

  OP.fit <- prodestOP(d$Y, fX = cbind(d$fX1, d$fX2), d$sX, d$inv, d$idvar, d$timevar)
OP.fit.solnp <- prodestOP(d$Y, fX = cbind(d$fX1, d$fX2), d$sX, d$inv, d$idvar, d$timevar, opt='solnp')
OP.fit.control <- prodestOP(d$Y, fX = cbind(d$fX1, d$fX2), d$sX, d$inv, d$idvar, d$timevar, cX = d$cX)

  # show results
  summary(OP.fit)
  summary(OP.fit.solnp)
  summary(OP.fit.control)

  # show results in .tex tabular format
   printProd(list(OP.fit, OP.fit.solnp, OP.fit.control))
```

`prodestWRDG` *Estimate productivity - Wooldridge method*

Description

The `prodestWRDG()` function accepts at least 6 objects (id, time, output, free, state and proxy variables), and returns a `prod` object of class `S4` with three elements: (i) a list of model-related objects, (ii) a list with the data used in the estimation and estimated vectors of first-stage residuals, and (iii) a list with the estimated parameters and their bootstrapped standard errors.

Usage

```
prodestWRDG(Y, fX, sX, pX, idvar, timevar, R = 20, G = 3, orth = F, cX = NULL, seed = 123456,
              tol = 1e-100, theta0 = NULL, cluster = NULL)
```

Arguments

`Y`	the vector of value added log output.
`fX`	the vector/matrix/dataframe of log free variables.
`sX`	the vector/matrix/dataframe of log state variables.
`pX`	the vector/matrix/dataframe of log proxy variables.
`cX`	the vector/matrix/dataframe of control variables. By default `cX= NULL`.
`idvar`	the vector/matrix/dataframe identifying individual panels.
`timevar`	the vector/matrix/dataframe identifying time.
`R`	the number of block bootstrap repetitions to be performed in the standard error estimation. By default `R = 20`.
`G`	the degree of the polynomial for productivity in sX and pX. By default, `G = 3`
`orth`	a Boolean that determines whether first-stage polynomial should be orthogonal or raw. By default, `orth = F`. It is recommended to set orth to T if degree of polynomial is high.
`theta0`	a vector with the second stage optimization starting points. By default `theta0 = NULL` and the optimization is run starting from the first stage estimated parameters + $N(\mu = 0, \sigma = 0.01)$ noise.
`cluster`	an object of class `"SOCKcluster"` or `"cluster"`. By default `cluster = NULL`.
`seed`	seed set when the routine starts. By default `seed = 123456`.
`tol`	optimizer tolerance. By default `tol = 1e-100`.

Details

Consider a Cobb-Douglas production technology for firm i at time t

- $y_{it} = \alpha + w_{it}\beta + k_{it}\gamma + \omega_{it} + \epsilon_{it}$

where y_{it} is the (log) output, w_it a 1xJ vector of (log) free variables, k_it is a 1xK vector of state variables and ϵ_{it} is a normally distributed idiosyncratic error term. The unobserved technical efficiency parameter ω_{it} evolves according to a first-order Markov process:

- $\omega_{it} = E(\omega_{it}|\omega_{it-1}) + u_{it} = g(\omega_{it-1}) + u_{it}$

and u_{it} is a random shock component assumed to be uncorrelated with the technical efficiency, the state variables in k_{it} and the lagged free variables w_{it-1}. Wooldridge method allows to jointly estimate OP/LP two stages jointly in a system of two equations. It relies on the following set of assumptions:

- a) $\omega_{it} = g(x_{it}, p_{it})$: productivity is an unknown function $g(.)$ of state and a proxy variables;
- b) $E(\omega_{it}|\omega_{it-1)} = f[\omega_{it-1}]$, productivity is an unknown function $f[.]$ of lagged productivity, ω_{it-1}.

Under the above set of assumptions, It is possible to construct a system gmm using the vector of residuals from

- $r_{1it} = y_{it} - alpha - w_{it}\beta - x_{it}\gamma - g(x_{it}, p_{it})$
- $r_{2it} = y_{it} - alpha - w_{it}\beta - x_{it}\gamma - f[g(x_{it-1}, p_{it-1})]$

where the unknown function $f(.)$ is approximeted by a n-th order polynomial and $g(x_{it}, m_{it}) = \lambda_0 + c(x_{it}, m_{it})\lambda$. In particular, $g(x_{it}, m_{it})$ is a linear combination of functions in (x_{it}, m_{it}) and c_{it} are the addends of this linear combination. The residuals eqnr_it are used to set the moment conditions

- $E(Z_{it} * r_{it}) = 0$

with the following set of instruments:

- $Z1_{it} = (1, w_{it}, x_{it}, c_{it})$
- $Z2_{it} = (w_{it-1}, c_{it}, c_{it})$

Value

The output of the function `prodestWRDG` is a member of the `S3` class **prod**. More precisely, is a list (of length 3) containing the following elements:

`Model`, a list containing:

- `method:` a string describing the method ('WRDG').
- `elapsed.time:` time elapsed during the estimation.
- `seed:` the seed set at the beginning of the estimation.
- `opt.outcome:` optimization outcome.

`Data`, a list containing:

- `Y:` the vector of value added log output.
- `free:` the vector/matrix/dataframe of log free variables.
- `state:` the vector/matrix/dataframe of log state variables.
- `proxy:` the vector/matrix/dataframe of log proxy variables.
- `control:` the vector/matrix/dataframe of log control variables.
- `idvar:` the vector/matrix/dataframe identifying individual panels.
- `timevar:` the vector/matrix/dataframe identifying time.

`Estimates`, a list containing:

- `pars:` the vector of estimated coefficients.
- `std.errors:` the vector of bootstrapped standard errors.

Members of class `prod` have an `omega` method returning a numeric object with the estimated productivity - that is: $\omega_{it} = y_{it} - (\alpha + w_{it}\beta + k_{it}\gamma)$. `FSres` method returns a numeric object with the residuals of the first stage regression, while `summary`, `show` and `coef` methods are implemented and work as usual.

Author(s)

Gabriele Rovigatti

References

Wooldridge, J M (2009). "On estimating firm-level production functions using proxy variables to control for unobservables." Economics Letters, 104, 112-114.

Examples

```
    data("chilean")

  # we fit a model with two free (skilled and unskilled), one state (capital) and one proxy variable (electricity)

   WRDG.fit <- prodestWRDG(d$Y, fX = cbind(d$fX1, d$fX2), d$sX, d$pX, d$idvar, d$timevar)

    # show results
    WRDG.fit

    ## Not run:
     # estimate a panel dataset - DGP1, various measurement errors - and run the estimation
      sim <- panelSim()

      WRDG.sim1 <- prodestWRDG(sim$Y, sim$fX, sim$sX, sim$pX1, sim$idvar, sim$timevar)
      WRDG.sim2 <- prodestWRDG(sim$Y, sim$fX, sim$sX, sim$pX2, sim$idvar, sim$timevar)
      WRDG.sim3 <- prodestWRDG(sim$Y, sim$fX, sim$sX, sim$pX3, sim$idvar, sim$timevar)
      WRDG.sim4 <- prodestWRDG(sim$Y, sim$fX, sim$sX, sim$pX4, sim$idvar, sim$timevar)

      # show results in .tex tabular format
    printProd(list(WRDG.sim1, WRDG.sim2, WRDG.sim3, WRDG.sim4), parnames = c('Free','State'))

## End(Not run)
```

---

prodestWRDG_GMM *Estimate productivity - Wooldridge method*

Description

The prodestWRDG_GMM() function accepts at least 6 objects (id, time, output, free, state and proxy variables), and returns a prod object of class S3 with three elements: (i) a list of model-related objects, (ii) a list with the data used in the estimation and estimated vectors of first-stage residuals, and (iii) a list with the estimated parameters and their bootstrapped standard errors.

Usage

```
prodestWRDG_GMM(Y, fX, sX, pX, idvar, timevar, G = 3, orth = F, cX = NULL, seed = 123456, tol = 1e-100)
```

Arguments

`Y`	the vector of value added log output.
`fX`	the vector/matrix/dataframe of log free variables.
`sX`	the vector/matrix/dataframe of log state variables.
`pX`	the vector/matrix/dataframe of log proxy variables.
`cX`	the vector/matrix/dataframe of control variables. By default `cX= NULL`.
`G`	the degree of the polynomial for productivity in sX and pX. By default, `G = 3`.
`orth`	a Boolean that determines whether first-stage polynomial should be orthogonal or raw. By default, `orth = F`. It is recommended to set orth to T if degree of polynomial is high.
`idvar`	the vector/matrix/dataframe identifying individual panels.
`timevar`	the vector/matrix/dataframe identifying time.
`seed`	seed set when the routine starts. By default `seed = 123456`.
`tol`	optimizer tolerance. By default `tol = 1e-100`.

Details

Consider a Cobb-Douglas production technology for firm i at time t

- $y_{it} = \alpha + w_{it}\beta + k_{it}\gamma + \omega_{it} + \epsilon_{it}$

where y_{it} is the (log) output, w_it a 1xJ vector of (log) free variables, k_it is a 1xK vector of state variables and ϵ_{it} is a normally distributed idiosyncratic error term. The unobserved technical efficiency parameter ω_{it} evolves according to a first-order Markov process:

- $\omega_{it} = E(\omega_{it}|\omega_{it-1}) + u_{it} = g(\omega_{it-1}) + u_{it}$

and u_{it} is a random shock component assumed to be uncorrelated with the technical efficiency, the state variables in k_{it} and the lagged free variables w_{it-1}. Wooldridge method allows to jointly estimate OP/LP two stages jointly in a system of two equations. It relies on the following set of assumptions:

- a) $\omega_{it} = g(x_{it}, p_{it})$: productivity is an unknown function $g(.)$ of state and a proxy variables;
- b) $E(\omega_{it}|\omega_{it-1)} = f[\omega_{it-1}]$, productivity is an unknown function $f[.]$ of lagged productivity, ω_{it-1}.

Under the above set of assumptions, It is possible to construct a system gmm using the vector of residuals from

- $r_{1it} = y_{it} - alpha - w_{it}\beta - x_{it}\gamma - g(x_{it}, p_{it})$
- $r_{2it} = y_{it} - alpha - w_{it}\beta - x_{it}\gamma - f[g(x_{it-1}, p_{it-1})]$

where the unknown function $f(.)$ is approximated by a n-th order polynomial and $g(x_{it}, p_{it}) = \lambda_0 + c(x_{it}, p_{it})\lambda$. In particular, $g(x_{it}, p_{it})$ is a linear combination of functions in (x_{it}, p_{it}) and c_{it} are the addends of this linear combination. The residuals eqnr_it are used to set the moment conditions

- $E(Z_{it} * r_{it}) = 0$

with the following set of instruments:

- $Z1_{it} = (1, w_{it}, x_{it}, c_{it})$
- $Z2_{it} = (1, w_{it-1}, x_{it}, c_{it-1}, q_{it})$

where q_{it-1} is a set of non-linear functions of c_it-1.

Value

The output of the function `prodestWRDG` is a member of the `S3` class **prod**. More precisely, is a list (of length 3) containing the following elements:

`Model`, a list containing:

- `method:` a string describing the method ('WRDG').
- `elapsed.time:` time elapsed during the estimation.
- `seed:` the seed set at the beginning of the estimation.
- `opt.outcome:` optimization outcome.

`Data`, a list containing:

- `Y:` the vector of value added log output.
- `free:` the vector/matrix/dataframe of log free variables.
- `state:` the vector/matrix/dataframe of log state variables.
- `proxy:` the vector/matrix/dataframe of log proxy variables.
- `control:` the vector/matrix/dataframe of log control variables.
- `idvar:` the vector/matrix/dataframe identifying individual panels.
- `timevar:` the vector/matrix/dataframe identifying time.

`Estimates`, a list containing:

- `pars:` the vector of estimated coefficients.
- `std.errors:` the vector of bootstrapped standard errors.

Members of class `prod` have an `omega` method returning a numeric object with the estimated productivity - that is: $\omega_{it} = y_{it} - (\alpha + w_{it}\beta + k_{it}\gamma)$. `FSres` method returns a numeric object with the residuals of the first stage regression, while `summary`, `show` and `coef` methods are implemented and work as usual.

Author(s)

Gabriele Rovigatti

References

Wooldridge, J M (2009). "On estimating firm-level production functions using proxy variables to control for unobservables." Economics Letters, 104, 112-114.

Examples

```
data("chilean")

# we fit a model with two free (skilled and unskilled), one state (capital) and one proxy variable (electricity)

WRDG.GMM.fit <- prodestWRDG_GMM(d$Y, fX = cbind(d$fX1, d$fX2), d$sX, d$pX, d$idvar, d$timevar)

# show results
WRDG.GMM.fit

# estimate a panel dataset - DGP1, various measurement errors - and run the estimation
sim <- panelSim()

WRDG.GMM.sim1 <- prodestWRDG_GMM(sim$Y, sim$fX, sim$sX, sim$pX1, sim$idvar, sim$timevar)
WRDG.GMM.sim2 <- prodestWRDG_GMM(sim$Y, sim$fX, sim$sX, sim$pX2, sim$idvar, sim$timevar)
WRDG.GMM.sim3 <- prodestWRDG_GMM(sim$Y, sim$fX, sim$sX, sim$pX3, sim$idvar, sim$timevar)
WRDG.GMM.sim4 <- prodestWRDG_GMM(sim$Y, sim$fX, sim$sX, sim$pX4, sim$idvar, sim$timevar)

# show results in .tex tabular format
printProd(list(WRDG.GMM.sim1, WRDG.GMM.sim2, WRDG.GMM.sim3, WRDG.GMM.sim4), parnames = c('Free','State'))
```

Index