Package ‘prodest’

October 14, 2022

Type Package
Title Production Function Estimation
Version 1.0.1
Author Gabriele Rovigatti [aut,cre]
Maintainer Gabriele Rovigatti <gabriele.rovigatti@gmail.com>

License GPL-3

BugReports https://github.com/GabrieleRovigatti/prodest/issues

URL https://github.com/GabrieleRovigatti/prodest/tree/master/prodest

LazyData TRUE

Depends R (>= 2.10), dplyr, parallel, Matrix, methods
Imports Rsolnp, DEoptim, AER
Suggests testthat

Repository CRAN
RoxygenNote 6.0.1

NeedsCompilation no

Date/Publication 2018-06-19 13:21:32 UTC

R topics documented:

block.boot.resample ........................................ 2
checkM ....................................................... 3
checkMD ...................................................... 3
chilean ........................................................ 4
coef ............................................................ 5
finalACF ...................................................... 5
block.boot.resample

Cluster Bootstrap Resampling

Description

Function to generate R vectors of resampled IDs. It works reshuffling the row number of the original data - which is stored in the input idvar along with the relative IDs. The output is a list (N_i x 1 x R), where N_i is a random number depending on the reshuffle.

Usage

block.boot.resample(idvar, R)

Arguments

idvar Vector of IDs to be resampled.
R Number of samples to be computed.

Details

block.boot.resample() accepts two inputs: a vector of IDs - i.e., the vector of panel identifier - and the number of resamplings. For each resampling, it reshuffles the IDs and outputs a vector whose row number is newly-created 'bootstrap' ID, while the value of each cell is the relative row to be reshuffled. This way, each individual can be sampled multiple times, keeping all her number of observations, without generating duplicates.
checkM

Author(s)
Gabriele Rovigatti

checkM Change input to matrix

Description
Function to transform all input to matrix.

Usage
checkM(input)

Arguments
input An R object. Can be a matrix/dataframe/vector/scalar.

Details
checkM() accepts one input and - if codeinput is a matrix - returns it without column names, otherwise transforms it into a matrix and returns it without column names.

Author(s)
Gabriele Rovigatti

checkMD Change dummy input to dummy matrix

Description
Function to transform all input to a matrix. In addition, it checks whether all elements of the input are either 0 or 1.

Usage
checkMD(input)

Arguments
input An R object. Can be a matrix/dataframe/vector/scalar.
chilean

Details

checkMD() accepts one input and - if codeinput is a matrix - returns it without column names, otherwise transforms it into a matrix and returns it without column names. In case any of the elements of input are different from 0 or 1, it stops the routine and throws an error.

Author(s)

Gabriele Rovigatti

---

chilean

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

Description


Usage

data("chilean")

Format

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

Value

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>vector of log(outcome) - Value added.</td>
</tr>
<tr>
<td>sX</td>
<td>vector of log(capital).</td>
</tr>
<tr>
<td>fX</td>
<td>matrix of log(skilled labor) and log(unskilled labor).</td>
</tr>
<tr>
<td>cX</td>
<td>vector of log(water).</td>
</tr>
<tr>
<td>pX</td>
<td>vector of log(electricity).</td>
</tr>
<tr>
<td>inv</td>
<td>vector of log(investment).</td>
</tr>
<tr>
<td>idvar</td>
<td>vector of panel identifier.</td>
</tr>
<tr>
<td>timevar</td>
<td>vector of time.</td>
</tr>
</tbody>
</table>

References

Description

This method provides the way to extract and print the estimated parameters from a prod S4 object - estimates from prodestOP, prodestLP, prodestACF, prodestWRDG and prodestWRDG_GMM - defined in the prodest package.

Usage

coef(object,...)

Arguments

object 

object of class prod.

...

Additional arguments.

Details

coeq accepts an S4 prod object and prints the vector of estimated parameters.

Author(s)

Gabriele Rovigatti

Description

finalACF is the function linking the data cleaning part of the routine with the final function to be bootstrapped.

Usage

finalACF(ind, data, fnum, snum, cnum, opt, theta0, boot = FALSE)
**Arguments**

- **ind**
  Vector of indices to reshuffle the data.

- **data**
  data.frame with the data to perform the estimation on.

- **fnum**
  Number of free variables.

- **snum**
  Number of state variables.

- **cnum**
  Number of control variables.

- **opt**
  String with the optimizer.

- **theta0**
  Vector of starting points.

- **boot**
  Binary indicator for the estimation routine being the baseline estimation (boot = FALSE, the default) or a bootstrap repetition.

**Details**

finalACF() accepts at least 7 inputs: a vector of reshuffled indices, the data.frame with the data, the number of free, state and control variables, the starting points and the optimizer. It collects the results of gACF() function - baseline and bootstrapped - calculates the standard errors and stores all in a prod object.

**Author(s)**

Gabriele Rovigatti

---

**finalOPLP**

*OP and LP estimation routine*

**Description**

finalOPLP is the function linking the data cleaning part of the routine with the final function to be bootstrapped.

**Usage**

finalOPLP(ind, data, fnum, snum, cnum, opt, theta0, boot, tol, att)

**Arguments**

- **ind**
  Vector of indices to reshuffle the data.

- **data**
  data.frame with the data to perform the estimation on.

- **fnum**
  Number of free variables.

- **snum**
  Number of state variables.

- **cnum**
  Number of control variables.

- **opt**
  String with the optimizer.

- **theta0**
  Vector of starting points.
FSres

**boot**
Binary indicator for the estimation routine being the baseline estimation or a bootstrap repetition.

**tol**
Optimization tolerance set.

**att**
Indicator for attrition in the data - i.e., if firms exit the market.

**Details**

`finalOPLP()` accepts at 9 inputs: a vector of reshuffled indices, the `data.frame` with the data, the number of free, state and control variables, the starting points, the optimizer, an indicator for bootstrapped repetitions and the optimization tolerance. It collects the results of `gACF()` function - baseline and bootstrapped - calculates the standard errors and stores all in a `prod` object.

**Author(s)**

Gabriele Rovigatti

---

**FSres**

*Generate the vector of the first stage residuals*

**Description**

This method provides the way to estimate the first stage residuals from a `prod` S4 object - estimates from `prodestOP`, `prodestLP`, `prodestACF`, `prodestWRDG` and `prodestWRDG_GMM` - defined in the `prodest` package

**Usage**

`FSres(object)`

**Arguments**

- `object` object of class `prod`.

**Details**

`FSres` accepts an S4 `prod` object and returns the vector of first stage residuals.

**Author(s)**

Gabriele Rovigatti
Description

gACF returns the second stage parameters estimates of ACF models. It is part of the prodestACF() routine.

Usage

gACF(theta, mZ, mW, mX, mlX, vphi, vlag.phi)

Arguments

theta Vector of parameters to be estimated.
mZ Matrix of instruments.
mW Weighting matrix.
mX Matrix of regressors.
mlX matrix of lagged regressors.
vphi Vector of fitted polynomial.
vlag.phi Lagged vector of fitted polynomial.

Details

gACF() estimates the second stage of ACF routine. It accepts 7 inputs, generates and optimizes over the group of moment functions E(xi_itZ^k_it).

Author(s)

Gabriele Rovigatti

Description

gOPLP returns the second stage parameters estimates of both OP and LP models. It is part of both prodestOP() and prodestsLP() routines.

Usage

gOPLP(vtheta, mX, mlX, vphi, vlag.phi, vres, stol, Pr.hat, att)
Arguments

- \textit{vtheta} Vector of parameters to be estimated.
- \textit{mX} Matrix of regressors.
- \textit{m1X} matrix of lagged regressors.
- \textit{vphi} Vector of fitted polynomial.
- \textit{vlag.phi} Lagged vector of fitted polynomial.
- \textit{vres} Vector of residuals of the free variables.
- \textit{stol} Number setting the tolerance of the routine.
- \textit{Pr.hat} Vector of fitted exit probabilities.
- \textit{att} Indicator for attrition in the data - i.e., if firms exit the market.

Details

gOPLP() estimates the second stage of OP and LP routines. It accepts 7 inputs, generates and optimizes over the group of moment functions \(E(e_{it}X^k_{it})\).

Author(s)
Gabriele Rovigatti

\begin{verbatim}
lagPanel(idvar, timevar, value)
\end{verbatim}

Description
Function to generate lagged variables in a panel.

Usage

\begin{verbatim}
lagPanel(idvar, timevar, value)
\end{verbatim}

Arguments

- \textit{idvar} vector of panel identifiers.
- \textit{timevar} vector of time identifiers.
- \textit{value} variable vector to be lagged.

Details

\texttt{lagPanel()} accepts three inputs (the ID, the time and the variable to be lagged) and returns the vector of lagged variable. Lagged inputs with no correspondence - i.e., \(X_{-1}\) - are returned as NA.

Author(s)
Gabriele Rovigatti
omega

Generate the omega estimates

**Description**

This method provides the way to estimate the omega residuals from a prod S4 object - estimates from prodestOP, prodestLP, prodestACF, prodestWRDG and prodestWRDG_GMM - defined in the prodest package.

**Usage**

omega(object)

**Arguments**

object  
object of class prod.

**Details**

omega accepts an S4 prod object and returns a vector of omega estimates.

**Value**

- A vector of productivity estimates - omega.

**Author(s)**

Gabriele Rovigatti

---

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)
panelSim

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), ∆ (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.

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 - σmeasurementerror = .1
- pX3 log proxy variable - σmeasurementerror = .2
- pX4 log proxy variable - σmeasurementerror = .5

Author(s)

Gabriele Rovigatti

References

Examples

```r
require(prodest)

## Simulate a dataset with 100 firms (T = 50).
## \code{PanelSim()} delivers the last 10% of usable time per panel.
panel.data <- panelSim(N = 100, T = 50)
attach(panel.data)

## Estimate various models
ACF.fit <- prodestACF(Y, fX, sX, pX2, idvar, timevar, theta0 = c(0.5, 0.5))
LP.fit <- prodestLP(Y, fX, sX, pX2, idvar, timevar)
WDRG.fit <- prodestWDRG(Y, fX, sX, pX3, idvar, timevar)

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

**printProd**

*Print output - 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**

```r
printProd(mods, modnames = NULL, parnames = NULL, outfile = NULL,
ptime = FALSE, nboot = FALSE, screen = 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`.
- `screen` print the table on-screen without TeX format. By default `screen = 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")

# run various models
WRDGfit <- prodestWRDG_GMM(chilean$Y, fX = cbind(chilean$fX1, chilean$fX2),
    chilean$sX, chilean$pX, chilean$idvar, chilean$timevar)
OPfit <- prodestOP(chilean$Y, fX = cbind(chilean$fX1, chilean$fX2), chilean$sX,
    chilean$pX, chilean$idvar, chilean$timevar)

# show the output in latex - tabular format
printProd(list(OPfit, WRDGfit), modnames = c("Olley-Pakes", "Wooldridge"),
    parnames = c("bunsk", "bsk", "bk"))
# show the output on-screen - no teX format
printProd(list(OPfit, WRDGfit), modnames = c("Olley-Pakes", "Wooldridge"),
    parnames = c("bunsk", "bsk", "bk"), screen = TRUE)

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 procedure:

- 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**

```r
prodestACF(Y, fX, sX, pX, idvar, timevar, R = 20, cX = NULL, opt = 'optim', 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 \(R = 20\).
- \(opt\): 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 \(cluster = NULL\).

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 \(\epsilon_{it}\) is a normally distributed idiosyncratic error term. The unobserved technical efficiency parameter \(\omega_{it}\) 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 \(w_{it-1}\). 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 \(\omega_{it}\);
- c) \(\omega_{it}\) 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 \(\epsilon_{it}\) from the output, \(y_{it}\). As in the OP/LP case, the inverted policy function replaces the productivity term \(\omega_{it}\) 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'.
- **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

Examples

```r
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_estadisticas_Enia.php

data(chilean)

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

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

set.seed(154673)
ACF.fit.solnp <- prodestACF(chilean$Y, fX = cbind(chilean$fX1, chilean$fX2), chilean$sX,
                            chilean$pX, chilean$idvar, chilean$timevar,
                            theta0 = c(.5,.5,.5), opt = 'solnp')

# run the same regression in parallel
# nCores <- as.numeric(Sys.getenv("NUMBER_OF_PROCESSORS")) # Windows systems
nCores <- 3
cl <- makeCluster(getOption("cl.cores", nCores - 1))
set.seed(154673)
ACF.fit.par <- prodestACF(chilean$Y, fX = cbind(chilean$fX1, chilean$fX2), chilean$sX,
                           chilean$pX, chilean$idvar, chilean$timevar,
                           theta0 = c(.5,.5,.5), cluster = cl)

stopCluster(cl)

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

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

---

**prodestLP**

*Estimate productivity - Levinsohn-Petrin method*

**Description**

The `prodestLP()` 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, cX = NULL,
        opt = 'optim', theta0 = NULL, cluster = NULL, tol = 1e-100, exit = FALSE)

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

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 param-
  eters + N(μ = 0, σ = 0.01) noise.

cluster
  an object of class "SOCKcluster" or "cluster". By default cluster = NULL.

tol
  optimizer tolerance. By default tol = 1e-100.

exit
  Indicator for attrition in the data - i.e., if firms exit the market. By default exit
  = FALSE; if exit = TRUE, an indicator function for firms whose last appearance
  is before the last observation’s date is generated and used in the second stage.
  The user can even specify an indicator variable/matrix/dataframe with the exit
  years.

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 \( \epsilon_{it} \) is a normally distributed idiosyncratic error term. The unobserved technical efficiency parameter \( \omega_{it} \) 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 \( w_{it-1} \). 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 \( \omega_{it} \);

c) \( \omega_{it} \) 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 \( \omega_{it} \) 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. Exploiting the residual \( \nu_{it} \) 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 \( \chi_{it} \) is an indicator 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’.
  - `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: \( \omega_{it} = y_{it} - (\alpha + w_{it} \beta + k_{it} \gamma) \). FRes 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


Examples

```r
require(prodest)

## Chilean data on production.
data(chilean)

# we fit a model with two free (skilled and unskilled), one state (capital)
# and one proxy variable (electricity)
set.seed(154673)
LP.fit <- prodestLP(chilean$Y, fX = cbind(chilean$fX1, chilean$fX2), chilean$sX, chilean$pX, chilean$idvar, chilean$timevar)
LP.fit.solnp <- prodestLP(chilean$Y, fX = cbind(chilean$fX1, chilean$fX2), chilean$sX, chilean$pX, chilean$idvar, chilean$timevar, opt = 'solnp')

## Not run:
# run the same model in parallel
require(parallel)
nCores <- as.numeric(Sys.getenv("NUMBER_OF_PROCESSORS"))
c <- makeCluster(getOption("cl.cores", nCores - 1))
set.seed(154673)
LP.fit.par <- prodestLP(chilean$Y, fX = cbind(chilean$fX1, chilean$fX2), chilean$sX, chilean$pX, chilean$idvar, chilean$timevar, cluster = c)
stopCluster(c)

## End(Not run)

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

---

**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**

```r
prodestOP(Y, fX, sX, pX, idvar, timevar, R = 20, cX = NULL, opt = 'optim', theta0 = NULL, cluster = NULL, tol = 1e-100, exit = FALSE)
```

**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`.
- **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`.
- **tol**: optimizer tolerance. By default `tol = 1e-100`.
- **exit**: indicator for attrition in the data - i.e., if firms exit the market. By default `exit = FALSE`; if `exit = TRUE`, an indicator function for firms whose last appearance is before the last observation's date is generated and used in the second stage. The user can even specify an indicator variable/matrix/dataframe with the exit years.
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 $\epsilon_{it}$ is a normally distributed idiosyncratic error term. The unobserved technical efficiency parameter $\omega_{it}$ 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 $w_{it-1}$. 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 $\omega_{it}$;
- c) $\omega_{it}$ 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 $\omega_{it}$ 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. Exploiting the residual $\epsilon_{it}$ 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 $\chi_{it}$ is an indicator 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'.
  - 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**


**Examples**

```r
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_estadisticas_enia.php

data(chilean)

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

OP.fit <- prodestOP(chilean$Y, fX = cbind(chilean$fX1, chilean$fX2), chilean$sX, chilean$inv, chilean$idvar, chilean$timevar)

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

OP.fit.control <- prodestOP(chilean$Y, fX = cbind(chilean$fX1, chilean$fX2),
                           chilean$sX, chilean$inv, chilean$idvar,
                           chilean$...)
```
chilean$timevar, cX = chilean$cX)

OP.fit.attrition <- prodestOP(chilean$Y, fX = cbind(chilean$fX1, chilean$fX2),
                           chilean$sX, chilean$inv, chilean$idvar,
                           chilean$timevar, exit = TRUE)

# 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, OP.fit.attrition))

---

prodestROB | Estimate productivity - Robinson-Wooldridge method

**Description**

The prodestROB() 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**

prodestROB(Y, fX, sX, pX, idvar, timevar, cX = 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.

**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 $\epsilon_{it}$ is a normally distributed idiosyncratic error term. The unobserved technical efficiency parameter $\omega_{it}$ 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, \( \omega_{it-1} \).

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

\[
\begin{align*}
  r_{1it} &= y_{it} - \alpha - w_{it}\beta - x_{it}\gamma - g(x_{it}, p_{it}) \\
  r_{2it} &= y_{it} - \alpha - w_{it}\beta - x_{it}\gamma - f[g(x_{it-1}, p_{it-1})]
\end{align*}
\]

where the unknown function \( f(.) \) is approximated 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 \( r_{it} \) are used to set the moment conditions

- \( E(Z_{it} r_{it}) = 0 \)

with the following set of instruments:

\[
\begin{align*}
  Z_{1it} &= (1, w_{it}, x_{it}, c_{it}) \\
  Z_{2it} &= (w_{it-1}, c_{it}, c_{it})
\end{align*}
\]

According to the input timing in ACF, the first equation proposed by Wooldridge would not be useful to identify any of the parameters, but it would be possible to achieve the identification from the second equation by exploiting the orthogonality condition:

- \( \epsilon_{it} | x_{it}, w_{it-1}, x_{it-1}, m_{it-1}, ..., x_{i1}, w_{i1}, m_{i1} ) = 0 \)

with an instrumental variable version of Robinson (1988)'s estimator.

Value

The output of the function prodestROB 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 ("ROB-IV").
- elapsed.time: time elapsed during 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

Examples
data("chilean")

# we fit a model with two free (skilled and unskilled), one state (capital)
# and one proxy variable (electricity)
ROB.IV.fit <- prodestROB(chilean$Y, fX = cbind(chilean$fX1, chilean$fX2),
                           chilean$sX, chilean$pX, chilean$idvar, chilean$timevar)

# show results
ROB.IV.fit

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

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

# show results in .tex tabular format
**prodestWRDG**

*Estimate productivity - IV 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 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**

```r
prodestWRDG(Y, fX, sX, pX, idvar, timevar, cX = 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.

**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 \( \epsilon_{it} \) is a normally distributed idiosyncratic error term. The unobserved technical efficiency parameter \( \omega_{it} \) 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 \( g(\cdot) \) of state and a proxy variables;
b) \( E(\omega_{it} | \omega_{it-1}) = f(\omega_{it-1}) \), productivity is an unknown function \( f(\cdot) \) of lagged productivity, \( \omega_{it-1} \).

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

\[
\begin{align*}
\quad r_{1it} &= y_{it} - \alpha - w_{it}\beta - x_{it}\gamma - g(x_{it}, p_{it}) \\
\quad r_{2it} &= y_{it} - \alpha - w_{it}\beta - x_{it}\gamma - f[g(x_{it-1}, p_{it-1})]
\end{align*}
\]

where the unknown function \( f(\cdot) \) is approximated 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 \( r_{it} \) are used to set the moment conditions

\[ E(Z_{it} * r_{it}) = 0 \]

with the following set of instruments:

\[
\begin{align*}
\quad Z_{1it} &= (1, w_{it}, x_{it}, c_{it}) \\
\quad Z_{2it} &= (w_{it-1}, c_{it}, c_{it})
\end{align*}
\]

Following previous assumptions, being \( f(\omega) = \delta_0 + \delta_1(c_{it}\lambda) + \delta_2(c_{it}\lambda)^2 + \ldots + \delta_G(c_{it}\lambda)^G \), one can set \( \delta_1 = G = 1 \) and estimate the model in a linear fashion - i.e., using a linear 2SLS model.

**Value**

The output of the function prodestWRDG is a member of the S3 class prod. More precisely, it 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.
  - `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

Examples

data("chilean")

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

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

# show results
WRDG.IV.fit

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

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

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

---

**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, cX = 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.
- **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 `\epsilon_{it}` is a normally distributed idiosyncratic error term. The unobserved technical efficiency parameter `\omega_{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, `\omega_{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}, 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 `r_{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.
- `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


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(chilean$Y, fX = cbind(chilean$fX1, chilean$fX2),
                                 chilean$sX, chilean$pX, chilean$idvar, chilean$timevar)
# show results

## estimate a panel dataset - DGP1, various measurement errors - and run the estimation

```r
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

```r
printProd(list(WRDG.GMM.sim1, WRDG.GMM.sim2, WRDG.GMM.sim3, WRDG.GMM.sim4),
          parnames = c('Free', 'State'))
```

---

**show**

*Print a table with parameter estimates*

**Description**

This method allows the user to print a table with the parameter estimates of an S4 prod object.

**Usage**

```r
show(object)
```

**Arguments**

- `object` object of class `prod`.

**Details**

- `show` accepts an S4 `prod` object and prints a table with estimated parameters.

**Author(s)**

Gabriele Rovigatti
**summary**

*Print a table with a summary of results*

**Description**

This method allows the user to print a table with a summary of the results within an S4 `prod` object: the parameter estimates, the method, the number of observations, the time elapsed, the number of bootstrap repetitions, the first stage estimates and the starting points.

**Arguments**

- `object`: object of class `prod`.
- `...`: Additional arguments.

**Details**

`summary` accepts an S4 `prod` object and prints a table with the results.

**Author(s)**

Gabriele Rovigatti

---

**weightM**

*Generate optimal GMM weighting matrix*

**Description**

In a Wooldridge estimation setting, i.e., in a system GMM framework, this function returns the optimal weighting matrix or the variance-covariance matrix given 1st or 2nd stage estimation results.

**Usage**

```r
weightM(Y, X1, X2, Z1, Z2, betas, numR, SE = FALSE)
```

**Arguments**

- `Y`: Vector of log(value added output).
- `X1`: Matrix of regressors for the first equation.
- `X2`: Matrix of regressors for the second equation.
- `Z1`: Matrix of instruments for the first equation.
- `Z2`: Matrix of instruments for the second equation.
- `betas`: Vector of first/second stage parameter estimates.
- `numR`: Number of state + number of free + number of control variables (i.e., number of constrained parameters).
- `SE`: Binary indicator for first (SE == FALSE, the default) or second stage.
Details

weightM() accepts at least 7 inputs: Y, X1, X2, Z1, Z2, betas and numR. With these, computes the optimal weighting matrix in a system GMM framework, i.e. $W^* = \sigma^*Z'Z$. If it is called during the first stage, it returns $W^*$, otherwise will return an estimate of the parameters’ standard errors, i.e., the square root of the diagonal of the variance-covariance matrix: $1/N( (X'Z) W^* (Z'X) )^{-1}$.

Author(s)

Gabriele Rovigatti

---

withinvar

Generate the variance of the demeaned variable

Description

withinvar() subtracts the mean of a vector from the vector itself, and then returns its variance.

Usage

withinvar(inmat)

Arguments

inmat Vector of log(value added output).

Details

withinvar() accepts a vector as input, then subtracts the mean from it and returns the variance.

Author(s)

Gabriele Rovigatti
Index

* classes
  prod, 13
* datasets
  chilean, 4

block.boot.resample, 2
checkM, 3
checkMD, 3
chilean, 4
coe, 5
coe, prod-method (coef), 5
data.frame, 4
finalACF, 5
finalOPLP, 6
FSres, 7
FSres, prod-method (FSres), 7
gACF, 8
gOPLP, 8

lagPanel, 9

omega, 10
omega, prod-method (omega), 10

panelSim, 10
printProd, 12
prod, 5, 7, 10, 13, 32, 33
prod-class (prod), 13
prodestACF, 5, 7, 10, 14
prodestLP, 5, 7, 10, 17
prodestOP, 5, 7, 10, 21
prodestROB, 24
prodestWRDG, 5, 7, 10, 27
prodestWRDG_GMM, 5, 7, 10, 29

show, 32
show, prod-method (show), 32

summary, 33
summary, prod-method (summary), 33
weightM, 33
withinvar, 34