Production Function Estimation in R: The prodest Package

Gabriele Rovigatti
Bocconi University

Abstract

This paper presents the R package **prodest** for production function estimation using the control function approach. Focusing on the Value Added PF, it provides functions to estimate two–steps models presented by Olley and Pakes (1996) and Levinsohn and Petrin (2003), as well as their correction proposed by Ackerberg et al. (2015). The system GMM framework proposed by Wooldridge (2009) is also implemented and tested in a series of Monte Carlo exercises. The **prodest** package features the DGP used by Ackerberg et al. (2015) and allows for the simulation of datasets according to various measurement errors and random shock variances. I illustrate the package use with an application to a popular firm–level dataset.

Keywords: production functions, productivity, prodest, R software .

1. Introduction

The correct estimation of the total factor productivity (TFP) is a fundamental issue in applied economics and the main topic of several seminal papers. On the one hand, when subject to positive productivity shocks, firms respond by both expanding their level of output and demanding more input; negative shocks, on the other hand, lead to a decline in output and demand for input. The positive correlation between the observable input levels and the unobservable productivity shocks is a major source of bias in OLS when estimating the total factor productivity. Various methods have been proposed to tackle such simultaneity issue and, according to their approaches, is possible to group them in three families: fixed effects, instrumental variables and control function. In the latter group, Olley and Pakes (1996) - OP henceforth - are the first to propose a two-step procedure aimed at overcoming the endogeneity: they use the investment level to proxy productivity. Their approach has been refined by Levinsohn and Petrin (2003) - LP - and Ackerberg et al. (2015) - ACF. Wooldridge (2009) proposes a novel estimation setting that shows i) how to obtain LP estimator within a system GMM econometric framework, which can be estimated in a single step, and ii) the appropriate moment conditions.

A crucial assumption underlies both the dynamic profit maximization problem faced by the firm at each time t and all the above models: the idiosyncratic shock to productivity at time t (i.e., ξ_t) does not affect the choice of the level of state variables, which is taken at $t - b$[1], but only that of free variables. Therefore, ξ_t is uncorrelated to the contemporaneous

[1] Where $b > 0$ can take different values depending on state variable dynamics.

value of the state and to all the lagged values of the free and state variables; all of these are valid instruments for parameter identification. These, in turn, can be used in GMM–type estimation settings.

prodest is a brand new package that implements - for the first time on R - all the main models proposed in the literature so far, in a user-friendly and intuitive way. Dealing with Value Added models, it allows users to perform (i) TFP estimation following OP, LP, ACF and Wooldridge methods, and (ii) simulation of production data according to the Data Generating Process firstly proposed by ACF. The package is written using `S4` classes, and provides several methods such as `coef()`, `summary()` and `show()`, alongside custom methods (`omega` and `FSres`) to extract and analyze the results.

The remainder of the paper is structured as follows: in Section 2 I review the control function approaches to TFP estimation in the literature, list their weaknesses and provide a general overview of the state of the art in empirical applications; Section 3 presents **prodest**, its main features, structure and functions; in Section 4 I present practical examples of the package usage, along with comparative results of the various models implemented on real and simulated data; Section 6 concludes.

2. Control function approach

For the remainder of the paper, consider a simple two-inputs Cobb-Douglas production function for firm i at time t:[2]

$$Y_{it} = A_{it} K_{it}^{\beta_k} L_{it}^{\beta_l} \tag{1}$$

where the output Y_{it} is a function of the two input "types" - capital K and labour L - and of the tecnical efficiency parameter A_{it}, typically unobservable. Taking the logs, (1) becomes

$$y_{it} = \beta_0 + l_{it}\beta_l + k_{it}\beta_k + \omega_{it} + \epsilon_{it} \tag{2}$$

where y_{it} is the log output - in most cases, it amounts to the value added - l_{it} is a $1 \times J$ vector of log free variable and k_{it} is a $1 \times K$ vector of log state variables, and $ln(A_{it}) = \beta_0 + \omega_{it}$ is the technical efficiency parameter - i.e., the parameter of interest - and ϵ_{it} is an output shock distributed as a white noise which is not observed by the firm *prior* to the exit or input level choices. In this sensethe idiosyncratic terms ω_{it} and ϵ_{it} are not identifiable in a linear regression setting, which would also suffer from a serious endogeneity (which, as outlined above, regards ω_{it} only), hence the need for different models. In particular, the literature has developed three main approaches to the estimation: Instrumental Variable, Fixed Effects and Control Function. The latter consists of structural empirical models aimed at performing the estimation through two-steps or single-step, system GMM procedures. The single most important assumption in such models is that the technical efficiency parameter dynamic follows a first-order Markov process like:

$$\omega_{it} = E(\omega_{it} \,|\Omega_{it-1}) + \xi_{it} = E(\omega_{it} \,|\omega_{it-1}) + \xi_{it} = g(\omega_{it-1}) + \xi_{it} \tag{3}$$

[2]The notation in this paper follows cosely that of Ackerberg et al. (2007) and the section structure recalls Mollisi and Rovigatti (2018).

where Ω stands for the information set and ξ_{it} is the shock to technical efficiency, orthogonal to ω_{it} and to the state variables in k_{it}.

2.1. Two-step estimators

OP, LP and ACF propose consistent two-step estimation procedures for (2). They rely on a few assumptions about both the timing of the underlying firms' production processes and the functional form of the variables proxying productivity - p_{it}. OP exploits firm investment levels as a proxy variable for ω_{it}, while LP and ACF propose to employ the intermediate inputs level - this approach being useful to overcome the empirical issue of zeros in investment data due to diverse managerial practices.[3]

OP and LP For both OP and LP methods, the assumptions needed to ensure consistency take the form:

A.1 $p_{it} = f(k_{it}, \omega_{it})$ is the proxy function, invertible and monotonically increasing in ω_{it};

A.2 The state variables level[4] evolve according to an investment policy function which is decided at time $t - b$, with $b \in [1, .., T]$;

A.3 The free variables[5] are non-dynamic, in the sense that their choice at t does not impact future profits, and are chosen at time t, *after* the firm productivity shock realizes.

Olley and Pakes (1996) show that, given A.1 and A.2, the investment function i_{it} is orthogonal to the state variable in a way that that $E[i_{it}|k_{it}] = 0$ and can be inverted, yielding a proxy for productivity of the form:

$$\omega_{it} = f^{-1}(i_{it}, k_{it}) = h(i_{it}, k_{it}) \tag{4}$$

that is, they propose to proxy the technical efficiency parameter ω_{it} by an unknown function $h(.)$ of observables. Rearranging (2) accordingly, one obtains a partially linear model, through which the parameters of the free variables - β_l - can be consistently estimated. In fact, plugging (4) in (2) yields

$$\begin{aligned} y_{it} &= \beta_0 + l_{it}\beta_l + k_{it}\beta_k + h(i_{it}, k_{it}) + \epsilon_{it} = \\ &= l_{it}\beta_l + \Phi_{it}(i_{it}, k_{it}) + \epsilon_{it} \end{aligned} \tag{5}$$

where $\Phi(i_{it}, k_{it}) = \beta_0 + k_{it}\beta_k + h(i_{it}, k_{it})$. The estimation is performed by a non-parametric approximation of $\Phi(i_{it}, k_{it})$ - i.e., a n^{th} order polynomial in i_{it} and k_{it} ($\Phi^n(\cdot)$)[6]. Treating $\Phi(\cdot)$ non-parametrically features some additional advantages: on the one hand, it does not require to specify the primitives of the model - indeed, the investment function $i(\cdot)$ is a function of

[3] Most firms accumulate resources even for many years, postponing investment, made all at once afterwards.

[4] Typically capital, k_{it}.

[5] Typically labour inputs, l_{it}.

[6] First Stage. Clearly, the polynomial approximation prevents from a correct estimation of state variable parameters, which must be retrieved in a second stage.

demand, market conditions, firm-specific characteristics, etc.) - nor, on the other hand, to properly characterize the complicated dynamic equilibrium of the game. The estimation of β_l strictly depends on A.3: the free variable parameters are correctly identified as long as labor is a non-dynamic input. Conversely, if it has dynamic implications, these would enter the investment function and $\Phi(\cdot)$.

The first stage yields an estimate $\hat{\Phi}_{it}$ of the term $\Phi(i_{it}, k_{it}) = \beta_0 + k_{it}\beta_k + h(i_{it}, k_{it})$. In turn, this means that, given a set of parameters $(\hat{\beta}_0, \hat{\beta}_k)$ is possible to retrieve the estimated technical efficiency parameter by just inverting the terms

$$\hat{\omega}_{it} = \hat{\Phi}_{it} - \hat{\beta}_0 - k_{it}\hat{\beta}_k \tag{6}$$

Once obtained $\hat{\beta}_l$, one can exploit the markovian nature of technical efficiency and rewrite model (5) for $y_{it} - l_{it}\hat{\beta}_l$ using (3):

$$\begin{aligned} y_{it} - l_{it}\hat{\beta}_l &= \beta_0 + k_{it}\beta_k + \omega_{it} + \epsilon_{it} = \\ &= \beta_0 + k_{it}\beta_k + g(\omega_{it-1}) + \xi_{it} + \epsilon_{it} = \\ &= \beta_0 + k_{it}\beta_k + g(\Phi_{it-1} - k_{it-1}\beta_k) + e_{it} \end{aligned} \tag{7}$$

with the residual $e_{it} = \xi_{it} + \epsilon_{it}$, uncorrelated by definition with all regressors and $g(.)$ can be approximated by a nth order polynomial. The model, then, can be estimated either by a non-linear least squares (NLLS) approach - which is the one originally proposed by Olley and Pakes (1996) - or by constructing a moment condition in state and estimated residuals $\hat{\xi}_{it}$ (as proposed by Ackerberg et al. (2015) and implemented in `prodest`).[7]

In particular, assuming $g(.)$ to follow a random walk, the second stage model reads

$$y_{it} - l_{it}\hat{\beta}_l = \beta_0 + (k_{it} - k_{it-1})\,\beta_k + \hat{\Phi}_{it-1} + e_{it} \tag{8}$$

and the estimation of equation (8) proceeds by setting up the moment conditions $E[e_{it}k_{it}] = 0$, $\forall k$, and the β_k* vector is the vector of parameters which minimizes the criterion function:

$$\beta_k* = \text{argmin}\left\{\sum_k \left(\sum_i \sum_t e_{it}k_{it}\right)^2\right\} \tag{9}$$

Levinsohn and Petrin (2003) remark that the OP approach has a major drawback in empirical applications: firm- or plant-level data typically have several zeros in investment data due to strategic planning practices which violate the monotonicity assumption A.1 and prevent the estimation of technical efficiency. In fact, the investment level is not decided at each point in time, but accumulated over the years and then made all at once. LP propose to overcome the issue by choosing as proxy variable for ω_{it} the intermediate input levels.

LP consider a slightly modified version of (2) accounting for the intermediate inputs (m_{it}) in the production function:

$$y_{it} = \beta_0 + l_{it}\beta_l + k_{it}\beta_k + \omega_{it} + m_{it}\beta_m + \epsilon_{it} \tag{10}$$

[7]In particular, the whole machinery boils down to non-parametrically regressing ω_{it} on ω_{it-1} by a non-parametric approximation, and estimate the residuals $\hat{\xi}_{it}$.

Under the set of assumptions A.1-A.3, and with the additional requirement that firms observe their productivity shock *prior* to deciding the level of the intermediate input, m_{it} is orthogonal to the state variables, hence $E[m_{it}|k_{it}] = 0$ and, similarly to $i(\cdot)$ in OP case, m_{it} can be inverted:

$$\omega_{it} = f^{-1}(m_{it}, k_{it}) = h(m_{it}, k_{it}) \tag{11}$$

which is an unknown function of observables. The estimation proceeds analogously to the OP case, by plugging (11) in (10) and treating $\Phi(\cdot)$ non-parametrically in the first stage.

$$\begin{aligned} y_{it} &= \beta_0 + l_{it}\beta_l + k_{it}\beta_k + m_{it}\beta_m + h(m_{it}, k_{it}) + e_{it} = \\ &= l_{it}\beta_l + \Phi_{it}(m_{it}, k_{it}) + e_{it} \end{aligned} \tag{12}$$

where $e_{it} = \xi_{it} + \epsilon_{it}$. As before, equation (12) is a partially linear model identified in the free variable vector but not in the proxy variable, m_{it}. At the true values $[\beta_k*, \beta_m*]$ we can define the residual function e_{it} like:

$$e_{it} = y_{it} - l_{it}\hat{\beta}_l - k_{it}\beta_k^* - g\left(\hat{\Phi}_{it-1}(\beta_m*) - k_{it-1}\beta_k\right) \tag{13}$$

However, given the correlation between m_{it} and ξ_{it} - since the intermediate input level is chosen after the technical efficiency shock realizes - e_{it} is no longer a combination of pure errors and the NLLS approach would yield inconsistent estimates. Hence, one should rely on a GMM estimator by setting the moment conditions $E[e_{it}z_{it}^k]$=0, $\forall k$, where k indexes the instrument vector $\mathbf{z} = [k_{it}, \mathbf{m}_{it-1}]$. In particular, the criterion reads

$$(\beta_k*, \beta_m*) = \text{argmax}\left\{\sum_k\left(\sum_i\sum_t e_{it}z_{it}^k\right)^2\right\} \tag{14}$$

consistently estimates the set of paramenters (β_k, β_m).

ACF correction Bond and Soderbom (2005) and Ackerberg et al. (2015) state that, according to assumptions A.1-A.3, the free variable coefficients could be consistently estimated in the first stage only if the free variables are orthogonal to the proxy variable. More specifically, LP assume that both labor and intermediate input are chosen simultaneously at t and, in turn, it means that both measures are function of the realized productivity shock and of the state variables in k_{it}. As a result, β_l would not be identifiable in the first stage due to the collinearity between l_{it} and m_{it}. In other terms:

$$\begin{aligned} m_{it} &= m(\omega_{it}, k_{it}) \\ l_{it} &= l(\omega_{it}, k_{it}) \end{aligned} \tag{15}$$

and, using the monotonicity condition A.2, ACF show that $l_{it} = l\left[h(m_{it}, k_{it}), k_{it}\right]$. Given that a similar collinearity issue affects the OP estimator, they propose an alternative estimation approach based on the distinction between labor input l_{it} and free variables w_{it}, and on the following set of (slightly) modified assumptions:

B.1 $p_{it} = p_{it}(k_{it},\, l_{it},\, \omega_{it})$ is the proxy variable policy function, invertible in ω_{it}. Moreover, p_{it} is monotonically increasing in ω_{it};

B.2 the state variables k_{it} are decided at time $t-b$;

B.3 the labor input, l_{it}, is chosen at time $t-\zeta$, where $0<\zeta<1$. The free variables, w_{it}, are chosen at time t when the firm productivity shock is realized;

B.4 the production function is value added in the sense that the intermediate input m_{it} does not enter the production function to be estimated.[8]

Under the set of assumptions B.1-B.3 the first stage estimation removes ε_{it} from the the output y_{it}. More specifically, the proxy variable can be inverted and plugged in equation (2) yielding:

$$y_{it} = \Phi_{it}(p_{it}, k_{it},\, w_{it},\, l_{it}) + \varepsilon_{it} \tag{16}$$

where $\Phi_{it}(p_{it}, k_{it},\, w_{it},\, l_{it}) = k_{it}\beta_k + w_{it}\beta_w + l_{it}\beta_l + h(p_{it}, k_{it},\, w_{it},\, l_{it})$. For any vector $(\beta_k^*,\, \beta_w^*,\, \beta_l^*)$, compute the estimated technical efficiency

$$\hat{\omega}_{it} = \hat{\Phi}_{it} - k_{it}\beta_k - w_{it}\beta_w - l_{it}\beta_l \tag{17}$$

and, similar to OP/LP models, exploit the Markov chain nature of technical efficiency as in (3) to set up the moment conditions $E[\xi_{it} z_{it}^k] = 0,\ \forall k$, where k indexes instruments $\mathbf{z} = (k_{it},\, m_{it-1},\, l_{it-1})$. Finally, the resulting GMM criterion reads (Second stage):

$$(\beta_k^*, \beta_l^*, \beta_m^*) = \text{argmin}\left\{ -\sum_k \left(\sum_i \sum_t \xi_{it} z_{it}^k \right)^2 \right\} \tag{18}$$

2.2. Single-step estimators

Wooldridge (2009) takes a different approach and addresses the endogeneity issue affecting (2) in a generalized method of moments (GMM) framework as in Wooldridge (1996). Such an approach has the advantages of i) overcoming the identification issue in LP method remarked by ACF and ii) yielding robust standard errors, possibly accounting for serial correlation and heteroskedasticity, as opposed to the bootstrapped SE required by OP and LP procedures.

In order to build the relevant moment restrictions, Wooldridge notes that, in equation (12), $\Phi(k_{it}, m_{it}) \equiv \beta_0 + k_{it}\beta_k + h(k_{it}, m_{it})$ and the assumption underlying the first stage in LP is:[9]

$$E(\epsilon_{it}|\omega_{it-1}, l_{it}, k_{it}, m_{it}, l_{it-1}, k_{it-1}, m_{it-1}, ..., l_{i1}, x_{i1}, m_{i1}) = 0 \tag{19}$$

with no functional form imposed on $h(\cdot)$. Moreover, the markovian nature of technical efficiency dynamics imposes the uncorrelation between shocks to productivity (ξ_{it}) and, on the

[8]The value added production function is required because Bond and Soderbom (2005) show that gross output production functions are not identifiable without imposing further restrictions of the model. See paragraph 4.1 of (Ackerberg et al. 2015) for the details.

[9]The same reasoning, with a slightly different notation, applies to the OP procedure.

one hand, contemporaneous values of state variables (k_{it}) and, on the other hand, past values of free variables and intermediate inputs (l_{it-1} and m_{it-1}). Hence, is possible to rewrite (3) like:

$$E(\omega_{it}|k_{it}, l_{it-1}, k_{it-1}, m_{it-1}, ..., l_{i1}, k_{i1}, m_{i1}) = E(\omega_{it}\,|\omega_{it-1}) = f[h(k_{it-1}, m_{it-1})] \tag{20}$$

and no functional form is imposed for $f(\cdot)$ as well. Equations (19) and (20) inform the moment restrictions to set up the system GMM:

$$y_{it} = \beta_0 + l_{it}\beta_l + k_{it}\beta_k + h(k_{it}, m_{it}) + \epsilon_{it} \tag{21}$$

$$y_{it} = \beta_0 + l_{it}\beta_l + k_{it}\beta_k + f[h(k_{it-1}, m_{it-1})] + \eta_{it} \tag{22}$$

where $\eta_{it} = \xi_{it} + \epsilon_{it}$. In practical applications, the unknown functional forms $h(\cdot)$ and $f(\cdot)$ are approximated by nth order polynomials in the state and proxy variables. In particular, if we assume that

$$h(k_{it},\, m_{it}) = \lambda_0 + c(k_{it}, m_{it})\lambda_1 \tag{23}$$

it implies $f(\omega_{it}) = \delta_0 + \delta_1[c(k_{it}, m_{it})\lambda_1] + \delta_2[k(x_{it}, m_{it})\lambda_1]^2 + ... + \delta_G[k(x_{it}, m_{it})\lambda_1]^G$, where G is the order of the polynomial approximating $f(\cdot)$. In `prodest`, and for the remainder of the paper, we consider the case with $G = 1$ and $\delta_1 = 1$. Provided that, a straightforward substitution in (21) and (22) yields

$$y_{it} = \alpha_0 + l_{it}\beta_l + k_{it}\beta_k + c(k_{it}, m_{it})\lambda_1 + \epsilon_{it} \tag{24}$$

$$y_{it} = \alpha_0 + l_{it}\beta_l + k_{it}\beta_k + c(k_{it-1}, m_{it-1})\lambda_1 + \eta_{it} \tag{25}$$

with $\alpha_0 = \beta_0 + \lambda_0$. For each $t > 1$ it is possible to setup an IV GMM based on the residual functions

$$\mathbf{r}_{it}(\theta) = \begin{pmatrix} r_{it1}(\theta) \\ r_{it2}(\theta) \end{pmatrix} = \begin{pmatrix} y_{it} - \zeta - l_{it}\beta - k_{it}\gamma - c(k_{it}, \mathbf{m}_{it})\lambda_1 \\ y_{it} - \theta - l_{it}\beta - k_{it}\gamma - c(k_{it-1}, \mathbf{m}_{it-1})\lambda_1 \end{pmatrix} \tag{26}$$

and the choice of instruments for (24) and (25) exploits the orthogonality conditions in (19)-(20). More specifically, Wooldridge defines

$$\mathbf{Z}_{it} = \begin{pmatrix} \mathbf{z}_{it1} \\ \mathbf{z}_{it2} \end{pmatrix} = \begin{pmatrix} (1, k_{it}, l_{it}, c(k_{it}, m_{it})) \\ (1, k_{it}, l_{it-1}, c(k_{it-1}, m_{it-1})) \end{pmatrix} \tag{27}$$

and the moment restrictions follow from $E[\mathbf{Z}'_{it}\mathbf{r}_{it}(\theta)] = 0$.

Following the remarks by ACF on input timing decisions, eq. (24) cannot identify any of the parameters of interest, however, as Wooldridge (2009) himself states, it is possible to semi-parametrically estimate eq. (25). In particular, provided the orthogonality condition

$$E(\eta_{it}|k_{it}, l_{it-1}, k_{it-1}, m_{it-1}, ..., l_{i1}, k_{i1}, m_{i1}) = 0, t = 2, .., T \tag{28}$$

an instrumental variable version of the estimator proposed by Robinson (1988) can achieve the identification of β_k and β_l.

3. The R package prodest

The R package **prodest** offers an integrated environment to deal with production function approach methods in R. The implementation is straightforward and follows from the steps and the models outlined in the previous section. In particular, the package includes functions for i) estimation, and ii) simulation of production function data along with a subset of a widely used dataset on Chilean industrial production.

The general nomenclature of estimation functions is "`prodest[method]()`" , where methods are *OP*, *LP*, *ACF*, *WRDG* and *ROB* (these refer to estimators proposed by (Olley and Pakes 1996; Levinsohn and Petrin 2003; Ackerberg et al. 2015; Wooldridge 2009; Robinson 1988), respectively). They all return objects of class "prod", featuring a *Model*, a *Data* and an *Estimates* lists. The function `panelSim()` is a Data Generating Process (DGP) returning a `data.frame` with 9 vectors: the ID variable, the time variable, the log output, the log of free and state variables, and four proxy variables with different values of measurement error. The DGP is directly taken from the one proposed and used by Ackerberg et al. (2015).

3.1. Data Structure and Specification

Production function estimation needs to be run on panel data, i.e., a specific type of data in which each row corresponds to an individual/time couple.[10] In **prodest**, estimation functions requrie the user to pass both an ID and a time variable: their combination must uniquely identify every observation and hence provide the panel dimensions of the data.

All **prodest** estimation functions accept at least 6 objects:[11]

- `Y`: the log value added output;
- `fX`: the log free variables. These often account for the labor input - l_{it} and w_{it} in previous models;
- `sX`: the log state variables. These often account for the capital input - k_{it} in previous models;
- `pX`: the log proxy variables. In OP these account for the amount of investment (corresponding to i_{it} in 5), whereas, following LP, ACF and WRDG, these account for the material input - m_{it};
- `idvar`: the id varaible identifying individual panels. More specifically, it must contain a unique identifier for the plant / firm, either in string or (preferred) numeric format;
- `timevar`: the time variable. It should contain the date (year, quarter, etc.) in numeric format.

[10] See Croissant and Millo (2008) for a discussion on panel data and panel data econometrics in R.

[11] These can be vectors, matrices or data.frame objects.

All objects must be of the same length - N^* - and can contain NAs and missing values. There is a number of optional input, mostly model-specific, which are accepted by most **prodest** functions. `cX` stores optional control variables used in the estimation, by default `cX = NULL`. The standard errors in the two-step models (OP, LP and the ACF-corrected versions) are computed by cluster bootstrap, hence there are further parameters related to bootstrap repetitions (`R`, which is 20 by default), second stage optimization (`opt`, `tol` and `theta0` to set the optimizer, the optimization tolerance and the starting points, respectively) and parallelization (`cluster`).

In particular, `opt` accepts a string element taking values *'optim'*, *'DEoptim'* or *'solnp'*. These refer to optimization procedures used for the second stage in the GMM framework: optim (package **stats**) performs general-purpose optimization based on Nelder-Mead algorithm; DE-optim - package **DEoptim** by (Mullen et al. 2011; Ardia et al. 2011) - implements the optimization by the Differential Evolution algorithmPrice *et al.* (2006); solnp - package **Rsolnp** by Ghalanos and Theussl (2015) - implements the solver originally proposed by Ye (1988).

The `cluster` option - `cluster = NULL` by default - accepts objects of class `SOCKcluster` or `cluster` (e.g., using `makeCluster()` in **parallel** by Tierney and R Core team).[12] When specified, the bootstrap repetitions for the second stage standard errors are performed in parallel. `theta0` is a numeric vector/matrix that can be specified in order to provide starting points in the optimization of the second stage. Its use is recommended for methods like ACF, which are very sensitive and whose value function appears to show several local maxima - section 4 for a general discussion of the issue. By default `theta0 = NULL` and, in this case, the starting points are the first-stage results plus a noise drawn from a standard normal ($\mu_0 = 0, \sigma^2 = 0.01$).

3.2. Data Transformation

Raw data passed to **prodest** are transformed and integrated in order to allow the estimation of the models described in the previous section. In order to approximate $h(\cdot)$, the routine generates a polynomial of 3^{rd} degree in state and proxy variables. Moreover, it generates lags of the state and free variables (k_{it-1} and l_{it-1}) to be used as instruments in the second stage estimation. Note that, in order to do that, the `lag()` function of package **stats** is not applicable, as it is unable to determine when individual changes. In addition, **prodest** works with unbalanced panels, and must deal with individual time series of different length.[13] The auxiliary function `lagPanel()` in **prodest** correctly handles the panel nature of the data and returns the lagged series.

Finally, all variables are collected in a `data.frame` object and are purged from missing values in both the original and the lagged variables - this is always compulsory, as the first observation *per panel* misses the lagged value.

3.3. Estimation

[12](Schmidberger et al. 2009) extensively present parallel computing in R.

[13]Balanced panel dataset are seldom observed in the producvitity estimation literature: in most cases, firms enter/exit the market at different rates, and this reflects in individual panels appearing or disappearing in the dataset.

Two-step estimators `prodestOP()` and `prodestLP()` implement the Olley and Pakes (1996) and Levinsohn and Petrin (2003) models, respectively. They proceed with a straightforward OLS estimation of eq. (5) and (12), where $\Phi(\cdot)$ is approximated by a 2^{nd} order polynomial in state and proxy variables (first stage). The regression yields consistent estimates of β_l and hence is possible to compute $\hat{\Phi}_{it} = \bar{y}_{it} - \hat{\beta}_l l_{it}$, where $\bar{y}_{it}$ stands for the first stage fitted values. The residuals $\epsilon^{fs}_{it} = y_{it} - \bar{y}_{it} = y_{it} - \hat{\beta}_0 - \hat{\beta}_l l_{it} - \beta^{\star}_k k_{it}$[14] are stored in the `FSresiduals` object. Using $\hat{\Phi}$ and the moment conditions in (9) or (14) the commands proceed with the GMM estimation of the second stage, which yields $\hat{\beta}_k$ by approximating $g(\cdot)$ through a 2^{nd} order polynomial. Standard errors for β_k, then, are computed by cluster-bootstrapping at the individual level the second stage; the standard errors of β_l are instead provided by the `lm()` command directly in the first stage.

According to Olley and Pakes (1996), the outlined model potentially suffers from a selection bias due to the nature of plant-level production data[15]. The fact that many individuals drop out of the sample is not surprising for practitioners since in many markets even very high levels of turnover among players are regularly observed. However, the choice to stop producing - i.e., to exit the market - is non-random and highly correlated with the unobservable technical efficiency parameter ω. Moreover, the straightforward solution "use a balanced sub-sample of plants" may introduce a different bias on factor coefficients. In fact, if we assume that plants with higher capital stocks are less likely to drop out of the sample in case of negative shocks - which is a reasonable point for several reasons - then reducing the observations on the survival rate basis would generate a sample with a non-zero correlation between ξ_{it} and k_{it} (i.e., would undermine the parameter identification). The solution proposed to correct for the attrition bias involves two steps. First, by estimating a probit of the survival binary variable on a polynomial of lagged values of state and proxy variables is possible to fit the survival probability at each point in time ($\hat{P}_{it}$). Then, a 3^{rd} order polynomial in $\hat{P}_{it}$ is used to augment the polynomial in eq. (**??**) and (13) and control for survival probability in the second stage:

$$e_{it} = y_{it} - l_{it}\hat{\beta}_l - k_{it}\beta^*_k - g\left(\hat{\Phi}_{it-1}\left(\beta_m *\right) - k_{it-1}\beta_k, \hat{P}_{it}\right) \tag{29}$$

In practice, this reduces to expand the polynomial in Φ with higher-order terms in $\hat{P}_{it}$ and all cross products and to minimize the resulting criterion.

`prodestACF()` builds on a similar framework, as remarked in the previous section, but the first stage regression in (16) does not yield the free variables' parameters $\hat{\beta}_l$, which is instead estimated jointly with $\hat{\beta}_k$ in the second stage regression. By exploiting the moment conditions in (**??**), the estimation is performed in a GMM setting similar to the one proposed by OP and LP. The standard errors of both the free and the state variables parameters are computed by cluster-bootstrapping at individual level the second stage.

Below I report an example of two-step estimators run on a subset of a well-known dataset of chilean production plants 1996-2006. The estimated model assumes a production function of

[14]Where $\star$ indicates the first-stage estimate, which is different from the "final" estimation $\hat{\beta}_k$ performed in the second stage.

[15]Which is the typical format of the data used for estimation.

the type

$$y_{it} = \beta_0 + skil_{it}\beta_{skil} + unskil_{it}\beta_{unskil} + k_{it}\beta_k + \omega_{it} + \epsilon_{it} \quad (30)$$

where *skil* and *unskil* are both free variables and stand for skilled and unskilled labor input, respectively. We use investment (OP) and material input (LP and ACF) to perform the above models. The ACF model Lastly, we print the results of all models using **prodest** auxiliary function `printProd()` - see section 5 for a detailed description.

```
R> require(prodest)
R> data(chilean)
R> OP <- prodestOP(Y, fX = cbind(fX1, fX2), sX, pX = inv, idvar, timevar, R = 20)
R> LP <- prodestLP(Y, fX = cbind(fX1, fX2), sX, pX, idvar, timevar, R = 20)
R> ACF <- prodestACF(Y, fX = cbind(fX1, fX2), sX, pX, idvar, timevar, R = 20,
                 + theta0 = (c(.5,.5,.5)))
R> printProd(list(OP,LP,ACF), modnames = c('OP','LP','Acf'),
                 + parnames = c('beta_sk','beta_unskil','beta_k'), screen = TRUE)
```

```
-- -- -- -- -- -- -- -- -- --
                OP     LP    Acf
-- -- -- -- -- -- -- -- -- --
 beta_sk       0.314  0.199  0.646
              (0.041) (0.026) (0.113)
------ ------ ------ ------ ------ ------
 beta_unskil  0.256  0.169  0.644
             (0.034) (0.019) (0.199)
------ ------ ------ ------ ------ ------
 beta_k       0.168  0.117  0.251
             (0.025) (0.048) (0.042)
------ ------ ------ ------ ------ ------
N             2544    2544    2544
------ ------ ------ ------ ------ ------
```

Single-step estimators The Wooldridge (2009) estimator is implemented in two versions by **prodest**, with functions `prodestWRDG()` and `prodestWRDG_GMM()`. In particular, the estimation performed by `prodestWRDG()` is obtained by approximating $f(\cdot)$ by a 1^{st} order polynomial (i.e., $G = 1$) and assuming $\delta_1 = 1$ in (22). In this case, the whole system can be written in the linear form

$$\mathbf{y}*_{it} = \mathbf{X}*_{it}\beta + \mathbf{r}_{it} \quad (31)$$

where $\mathbf{y}*_{it}$ is a vector of twice stacked y_{it}, β is the vector of parameters of interest, $\mathbf{r}_{it}$ is the vector of residuals - more specifically, it is obtained stacking the residual vectors in 26. The matrix of regressors $\mathbf{X}*_{it}$ is defined as

$$\mathbf{X}_{it} = \begin{pmatrix} 1 & 0 & \mathbf{l}_{it} & \mathbf{k}_{it} & c(\mathbf{k}_{it}, m_{it}) \\ 0 & 1 & \mathbf{l}_{it} & \mathbf{k}_{it} & c(\mathbf{k}_{it-1}, m_{it-1}) \end{pmatrix} \quad (32)$$

and the estimation proceeds with a plain linear IV estimation using the instruments $\mathbf{Z}_{it}$ as in (27). In **prodest** the estimation is run through a 2SLS approach with `lm()`, and the standard errors are computed accordingly.

`prodestWRDG_GMM()` implements a slightly different version of the same model aimed at the estimation within a GMM framework. Stacking variables in eq. (24) and (25) would generate a matrix suffering issues due to the collinearity of regressors. In order to address the issue, the stacked matrix $\mathbf{X}*_{it}$ is built in the following way:

$$\mathbf{X}*_{it} = \begin{bmatrix} X1_1^1 & \cdots & X1_1^k & X1_1^{k+1} & \cdots & X1_1^{r1} & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & 0 & 0 & 0 \\ X1_N^1 & \cdots & X1_N^k & X1_N^{k+1} & \cdots & X1_N^{r1} & 0 & 0 & 0 \\ X2_1^1 & \cdots & X2_1^k & 0 & 0 & 0 & X2_1^{k+1} & \cdots & X2_1^{r2} \\ \vdots & \ddots & \vdots & 0 & 0 & 0 & \vdots & \ddots & \vdots \\ X2_N^1 & \cdots & X2_N^k & 0 & 0 & 0 & X2_N^{k+1} & \cdots & X2_N^{r2} \end{bmatrix} \tag{33}$$

where k is the number of regressors common to both equations - i.e., the constant β_0, the free and state variables in equations (21) and (21) - $r1$ the number of regressors in the first equation only and $r2$ the regressors in the second. The estimation is then performed in two steps: first, the GMM solution for over-identified linear models reads $\hat{\beta}^0 = ((X^{*\prime}Z)W^0(Z'X^*))^{-1}(X^{*\prime}Z)W^0(Z'Y^*)$ - first step with unadjusted and independent weighting matrix W^0. Using $\hat{\beta}^0$, define the optimal weighting matrix $W^\star = \sigma_{rs}Z'Z$ and the final parameters are estimated in the second step like $\hat{\beta}^\star = ((X^{*\prime}Z)W^\star(Z'X^*))^{-1}(X^{*\prime}Z)W^\star(Z'Y^*)$. The variance-covariance matrix is $Var(\hat{\beta}^\star) = \frac{1}{N}((X^{*\prime}Z)W^\star(Z'X^*))^{-1}$.[16]

prodest implements the semi-parametric estimator proposed by Wooldridge (2009) - which is the IV version of the estimator originally proposed by Robinson (1988) in a different context. More specifically, it is possible to identify the parameters in(22) by using the lag of the free variables as excluded instruments. In `prodestROB()`, the estimation is performed through the `ivreg` function in package **AER** by Kleiber and Zeileis (2008).

```
R> require(prodest)
R> data(chilean)
R> WRDG <- prodestWRDG(Y, fX = cbind(fX1, fX2), sX, pX = inv, idvar, timevar)
R> WRDG.GMM <- prodestWRDG_GMM(Y, fX = cbind(fX1, fX2), sX, pX, idvar, timevar)
R> ROB <- prodestROB(Y, fX = cbind(fX1, fX2), sX, pX, idvar, timevar)
R> printProd(list(WRDG, WRDG.GMM, ROB), modnames = c('WRDG','WRDG-GMM','ROB'),
                 + parnames = c('beta_sk','beta_unskil','beta_k'), screen = TRUE)
```

```
-- -- -- -- -- -- -- -- -- --
             WRDG  WRDG-GMM  ROB
-- -- -- -- -- -- -- -- -- --
 beta_sk      0.375  0.247  0.255
             (0.015) (0.017) (0.02)
------ ------ ------ ------ ------ ------ ------
 beta_unskil 0.309  0.217  0.218
```

[16] see (Wooldridge 2001, 2002)

```
            (0.014) (0.015) (0.018)
------ ------ ------ ------ ------ ------ ------
 beta_k       0.18  0.058  0.16
           (0.028) (0.031) (0.025)
------ ------ ------ ------ ------ ------ ------
N            2544    1944    2544
------ ------ ------ ------ ------ ------ ------
```

Output objects class All **prodest** estimation functions return objects of the `S3` class `prod`. These are lists of length 3 with elements `Model`, `Data` and `Estimates`; in turn, each element features several subelements:

- `Model`:
  - `$method`, a string with the method - OP, LP, WRDG, WRDG.GMM, ROB - used in the estimation;
  - `$boot.repetitions`, the number of boostrap repetitions used to compute the standard errors (OP, LP and ACF only);
  - `$elapsed.time`, the estimation time;
  - `$theta0`, the vector of optimization starting points (OP, LP and ACF only);
  - `$opt`, a string with the optimizer used in the second stage (OP, LP and ACF only);
  - `$opt.outcome`, the output from the optimizer (OP, LP and ACF only);
  - `$FSbetas`, the vector of first stage estimated parameters (OP, LP and ACF only).
- `Data`:
  - `$Y`, the log value added;
  - `$free`, `$state`, `$proxy`, `$control`, the log free, state, proxy and (optional) control variables;
  - `$idvar` stores the panel variable - i.e., the variable with the individual identifier;
  - `$timevar` stores the time variable;
  - `$FSresiduals` vector contains the vector of first-stage residuals.
- `Estimates`:
  - `$pars`, the named vector of estimated parameters;
  - `$std.errors`, the named vector of parameters' standard errors.

3.4. Simulation

Ackerberg et al. (2015) in their seminal paper, discuss in-depth the Data Generating Process underlying production functions and, in particular, focus on the *functional dependence* problem affecting the identification of β_l in the LP procedure.[17] They consider, for their Monte Carlo exercises, three different DGPs. First, they allow for serial correlation in wages with labor input decisions taken at t (DGP1); second, they implement a model with optimization error in the labor input decision (DGP2); third, both sources of heterogeneity are considered at once (DGP3). Finally, in all models they allow the input variables to be measured with errors; more specifically, for each DGP, they run models with $\sigma_{error} = 0$, .1, .2 or .5. `panelSim()` implements all the above DGPs, allowing the user to specify the number of "firms", the simulated times and several other characteristics.

`panelSim()` accepts up to 9 optional inputs, namely

- `N`, which is the number of firms;
- `T`, the time spanned by the simulation. Only the last 10% of simulated times *per panel* will be returned;
- `alphaL` and `alphaK`, the parameters of the free and the state variables, respectively;
- `DGP`, which controls for the type of DGP;
- `rho` is the AR(1) coefficient of ω in equation (3);
- `sigeps` and `sigomg`, the standard deviations of ϵ and of the productivity shocks;
- `rholnw`, which is the AR(1) coefficient of log(wages)

`panelSim()` returns a `data.frame` with the `idvar`, the `timevar`, the output `Y`, the free - `fX` - and state - `sX` - variables. In addition, it produces 4 proxy variables (`pX1`-`pX4`), simulated with a measurement error drawn from distributions with standard deviation of 0, 0.1, 0.2 and 0.5, respectively. `timevar` will amount to the 10% of `T`, in order to deliver only observations simulated in equilibrium.

By default, `N = 1,000` and `T = 100`, hence `panelSim()` returns a $N \times (T/10) = 1,000 * 10 = 10,000$ observations `data.frame` with 9 variables. Below a usage example (Monte Carlo with ACF and LP models) with `N = 500` and `T = 100`, repeated for all possible values of σ_{error}.

```
R> require(prodest)
R> prod.data <- panelSim(N = 500)
R> attach(prod.data)
R> ACF.fit1 <- prodestACF(Y, fX, sX, pX1, idvar, timevar, theta0 = c(.5,.5))
R> LP.fit1 <- prodestLP(Y, fX, sX, pX1, idvar, timevar)
R> ACF.fit2 <- prodestACF(Y, fX, sX, pX2, idvar, timevar, theta0 = c(.5,.5))
R> LP.fit2 <- prodestLP(Y, fX, sX, pX2, idvar, timevar)
R> ACF.fit3 <- prodestACF(Y, fX, sX, pX3, idvar, timevar, theta0 = c(.5,.5))
```

[17]In a nutshell, if l_{it} is functionally dependent from k_{it}, m_{it} and t - in the sense that l_{it} is *only* a function of capital, intermediate input and time - then the first stage regression is not identifiable, because there is no source of variation in l_{it} conditional on the above elements. Indeed, they show that LP procedure in the first stage correctly identifies the parameter of free variables only in three - very specific - cases: i) there is an i.i.d. optimization error in the free variables - and not in the proxy, ii) i.i.d. shock to the price of either labor or output at $t - \epsilon$, when the level of proxy variable has already been chosen, but not that of the free variables, iii) labor is chosen at $t - b$, as a function of ω_{t-b}, but the investment level is chosen at t.

```
R> LP.fit3 <- prodestLP(Y, fX, sX, pX3, idvar, timevar)
R> ACF.fit4 <- prodestACF(Y, fX, sX, pX4, idvar, timevar, theta0 = c(.5,.5))
R> LP.fit4 <- prodestLP(Y, fX, sX, pX4, idvar, timevar)
R> printProd(list(LP.fit1, ACF.fit1,LP.fit2, ACF.fit3,LP.fit2, ACF.fit3,LP.fit4,
+ ACF.fit4), screen = TRUE, modnames = c("LP1", "ACF1","LP2", "ACF2","LP3",
+ "ACF3","LP4", "ACF4"), parnames = c("beta_free", "beta_state"))
```

```
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
             LP1     ACF1     LP2    ACF2     LP3    ACF3     LP4   ACF4
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 beta_free   0.001   0.601   0.679   0.638   0.679   0.638   0.875   0.654
            (0.008) (0.013) (0.008) (0.017) (0.008) (0.017) (0.006) (0.02)
---------------- ---------------- ---------------- ---------------- ----------------
 beta_state  1.072   0.385  0.356   0.382   0.356   0.382   0.137    0.36
            (0.047) (0.02) (0.014) (0.022) (0.014) (0.022) (0.139) (0.028)
---------------- ---------------- ---------------- ---------------- ----------------
N            5000    5000    5000    5000    5000    5000    5000    5000
---------------- ---------------- ---------------- ---------------- ----------------
```

3.5. Methods

prodest features several post-estimation methods aimed at handling `prod` objects. Among them, `show()`, `coef()`, `FSres()` and `summary`, aimed at extracting parts of the stored results; see `help("prod")`. `omega()` generates the residuals of the second stage, namely, the estimates of the log productivity term.

4. Comparative results

4.1. Real Data

In table (1) we report the results of all models implemented in **prodest** on a sectoral subset of the well-known and broadly used dataset of Chilean firms 1986-1996.

As an illustration of how **prodest** works, the code producing table (1) reads

```
R> require(prodest)
R> data(chilean)
R> OP <- prodestOP(Y, fX = cbind(fX1, fX2), sX, pX = inv, idvar, timevar, R = 100)
R> LP <- prodestLP(Y, fX = cbind(fX1, fX2), sX, pX, idvar, timevar, R = 100)
R> ACF <- prodestACF(Y, fX = cbind(fX1, fX2), sX, pX, idvar, timevar, R = 100)
R> WRDG <- prodestWRDG(Y, fX = cbind(fX1, fX2), sX, pX, idvar, timevar)
R> WRDG.GMM <- prodestWRDG_GMM(Y, fX = cbind(fX1, fX2), sX, pX, idvar, timevar)
R> printProd(list(OP,LP,ACF,WRDG,WRDG.GMM), modnames = c('OP','LP','Acf','Wrdg'
                        + ,'Wrdg.GMM'), parnames = c('$\\beta_{skil}$',
                        + '$\\beta_{unskil}$','$\\beta_{k}$'))
```

Table 1: Chilean dataset - sectoral estimation

	OP	LP	Acf	Wrdg	Wrdg.GMM
β_{skil}	0.314	0.199	0.154	0.246	0.247
	(0.041)	(0.025)	(0.152)	(0.014)	(0.017)
β_{unskil}	0.256	0.169	0.158	0.214	0.217
	(0.032)	(0.024)	(0.134)	(0.013)	(0.015)
β_k	0.168	0.117	0.138	0.145	0.058
	(0.031)	(0.044)	(0.092)	(0.024)	(0.031)
N	2544	2544	2544	2544	1944

Note: productivity estimation on a sector-specific sample of Chilean firms 1986-1996. Free variables are *skilled* and *unskilled* quantities of labor input, the state variable is capital k and the proxy variables is investment - OP - or material inputs - LP, ACF, WRDG. Column titles indicate the estimated models. Standard errors are bootstrapped - for all but *WRDG* and *Wrdg.GMM* models - and reported in parenthesis.

```
\begin{tabular}{ccccccccccc}\hline\hline
 & & OP & & LP & & Acf & & Wrdg & & Wrdg.GMM \\\hline
 $\beta_{skil}$ & & 0.314 & & 0.199 & & 0.154 & & 0.246 & & 0.247 \\
 & & (0.041) & & (0.025) & & (0.152) & & (0.014) & & (0.017) \\
 &  &  &  &  &  \\
 $\beta_{unskil}$ & & 0.256 & & 0.169 & & 0.158 & & 0.214 & & 0.217 \\
 & & (0.032) & & (0.024) & & (0.134) & & (0.013) & & (0.015) \\
 &  &  &  &  &  \\
 $\beta_{k}$ & & 0.168 & & 0.117 & & 0.138 & & 0.145 & & 0.058 \\
 & & (0.031) & & (0.044) & & (0.092) & & (0.024) & & (0.031) \\
 &  &  &  &  &  \\
 &  &  &  &  &  \\
N & & 2544 & & 2544 & & 2544 & & 2544 & & 1944 \\\hline\hline
\end{tabular}
```

4.2. Monte Carlo simulations

As remarked above, using `panelSim()` allows the user to run Monte Carlo simulations on data simulated using the DGPs proposed by Ackerberg et al. (2015). In table (2) we report a replica of Ackerberg et al. (2015)'s Table I using `prodestACF()` - columns 1-4 - and `prodestLP()` - columns 5-8 - on 12 different simulated datasets (DGP1-3, each featuring 4 different levels of measurement error). Results replicate fairly well those presented in the original paper.

Table 2: ACF and LP - Monte Carlo Simulations

	ACF				LP			
	β_l		β_k		β_l		β_k	
Meas. Error	Coeff.	St. Dev.	Coeff.	St. Dev.	Coeff.	St. Dev.	Coeff.	St. Dev.
DGP1 - Serially Correlated Wages and Labor Set at Time $t-b$								
0.0	0.599	0.009	0.401	0.015	-0.004	0.005	1.099	0.030
0.1	0.600	0.011	0.425	0.016	0.679	0.009	0.365	0.012
0.2	0.621	0.012	0.406	0.015	0.790	0.007	0.242	0.010
0.5	0.668	0.015	0.357	0.017	0.876	0.005	0.195	0.160
DGP2 - Optimization Error in Labor								
0.0	0.606	0.054	0.395	0.054	0.600	0.003	0.401	0.013
0.1	0.610	0.030	0.406	0.033	0.755	0.004	0.257	0.009
0.2	0.617	0.021	0.404	0.024	0.809	0.004	0.205	0.010
0.5	0.633	0.019	0.389	0.022	0.864	0.003	0.446	0.238
DGP3 - Optimization Error in Labor and Serially Correlated Wages and Labor Set at Time $t-b$ (DGP1 plus DGP2)								
0.0	0.593	0.005	0.409	0.014	0.473	0.003	0.576	0.017
0.1	0.602	0.037	0.425	0.041	0.635	0.005	0.417	0.012
0.2	0.610	0.042	0.424	0.041	0.702	0.005	0.348	0.012
0.5	0.621	0.023	0.415	0.027	0.778	0.005	1.320	0.191

Notes: 1000 replications. True values of parameters are $\beta_l = 0.6$ and $\beta_k = 0.4$. Standard deviations have been calculated among 1000 replications. ρ is set at .7 and we used **optim** (optimizer: 'BFGS').
DGP1 - Serially Correlated Wages and Labor Set at Time $t-b$
DGP2 - Optimization Error in Labor
DGP3 - Optimization Error in Labor and Serially Correlated Wages and Labor Set at Time $t-b$ (DGP1 plus DGP2)

5. Auxiliary Functions

5.1. lagPanel() and printProd()

`lagPanel()` is an auxiliary function used to obtained lagged values of variables, individual by individual, in panel datasets. The function accepts 3 inputs - namely the `idvar`, the `timevar` and the `variable` to be lagged - and returns a vector storing the lagged values of the original `variable`. The first observation per individual is missing - being the lagged value of the first observations, unavailable by definition.

`printProd()` accepts a list of `prod` objects and up to 6 optional inputs. By default, it prints on screen a .tex tabular with the estimation results of the models passed to the function.

- `mods` is a list of `prod` objects - i.e., output from **prodest** estimation functions;
- `modnames` optionally provides the labels of the models in `mods`. By default `modnames = NULL`;
- `parnames` optionally stores the names of the reported parameters. By default `parnames = NULL`;
- `outfile` is the name of the file (in .tex format) which stores the tabular produced by `printProd()`. By default `outfile = NULL`;
- `ptime` optionally adds the estimation time to the output table. By default `ptime = FALSE`;
- `nboot` optionally adds the number of bootstrap repetitions to the output table. By default `nboot = FALSE`;
- `screen` prints the output results on screen. By default `screen = FALSE`.

5.2. Parallel

Parallel computing is extremely useful in **prodest** applications, mostly when dealing with a high number of bootstrap repetitions. In table (3) we report the results, in terms of estimates, number of bootstrap repetitions and computing time, of the ACF model run on a simulated dataset (`DGP = 2`, `N = 1000`, `T = 100`) with 100 and 1000 bootstrap repetitions and estimated plainly - columns (1) and (4) - or parallelized over 2 - columns (2) and (5) - and 3 - columns (3) and (6) - cores. See *Computational Details* below for a full reference to the hardware used.

As expected, the computational time decreases with the number of cores employed, and it is particularly relevant when the number of bootstrap repetitions increases: with 1000 repetitions, using 3 cores halves the computational time (≈ 6 to ≈ 3 mins).

6. Conclusions

This article introduces the R package **prodest** for production function estimation using the control function approach. **prodest** is the first R package that implements the models proposed by Olley and Pakes (1996), Levinsohn and Petrin (2003), Ackerberg et al. (2015)

Table 3: ACF estimation in parallel - various cores

	100 Repetitions			1000 Repetitions		
	Acf	Acf-par2	Acf-par3	Acf	Acf-par2	Acf-par3
β_{free}	0.617	0.617	0.617	0.617	0.617	0.617
	(0.007)	(0.007)	(0.007)	(0.009)	(0.009)	(0.009)
β_{state}	0.386	0.386	0.386	0.386	0.386	0.386
	(0.014)	(0.014)	(0.014)	(0.016)	(0.016)	(0.016)
Time	0.59	0.48	0.44	6.19	3.43	3.23
BootRep	100	100	100	1000	1000	1000
N	10000	10000	10000	10000	10000	10000

Notes: all models are estimated on a DGP1, `N=1000`, `T=100` simulated dataset. Columns (1)-(3) show models estimated with 100 bootstrap repetitions for the standard errors, while (4)-(6) employ 1000 repetitions. Estimates are obtained using 1 (plain), 2, or 3 cores in parallel.

and Wooldridge (2009); as such, it allows researchers to perform their applied research on TFP within the `R` environment. Among the main features of the package, the `panelSim()` functions provides a powerful tool for practitioners interested in running Monte Carlo, being able to simulate data from DGPs with diverse characteristics, and already used in the relevant literature.

The estimation functions deal with the panel data structure in a relatively simple way, as users are required to specify the `idvar` and `timevar` at each call. On the input side, all functions require the free, state and proxy variables either as `data.frame` or `matrix` objects. Moreover, they feature options to take care of model characteristics (i.e., attrition, control variables, etc.) and optimization procedures (e.g., starting points, choice of optimizer, tolerance). On the output side, the new class `prod` objects store all the relevant information on the model, the data and the estimated parameters; moreover, **prodest** includes a series of auxiliary functions and methods sepcifically developed to deal with these objects.

Computational Details

All results in this paper were obtained using R 3.3.2 (R Core Team 2017) with the packages: **prodest** version 1.0.1 Rovigatti (2017), **Rsolnp** version 1.16 Ghalanos and Theussl (2015), **DEoptim** version 2.2.4 Ardia et al. (2011), **plyr** version 1.8.4 Wickham (2011), **parallel** version 3.3.2 R Core Team (2017), **Matrix** version 1.2.7.1 Bates and Maechler (2016). Computations were performed on a Intel® Core CPU i7–6500U 2.59Ghz processor.

R itself and all packages used are available from CRAN at `http://CRAN.R-project.org/`. The package **prodest** is available from the CRAN repository at `https://cran.r-project.org/package=prodest`. The version under development is available in GitHub at `https://github.com/GabrieleRovigatti/prodest`.

Acknowledgments

Special thanks go to Francesco Decarolis who provided expertise and support. Also, I thank Daniel Ackerberg, Federico Belotti, Sara Calligaris, Leopoldo Catania, Vincenzo Mollisi, Andrea Pozzi, Lorenzo Rovigatti and Federico Tullio for comments that improved the package and the paper. Any errors are my own.

References

Ackerberg, D. and Caves, K. and Frazer, G. (2015). "Econometric tools for analyzing market outcomes." *Handbook of econometrics*, **6**, 2411–2451.

Ackerberg, D. and Caves, K. and Frazer, G. (2015). "Identification Properties of Recent Production Function Estimators." *Econometrica*, **83**(6), 2411–2451.

Ardia, D. and Mullen, K. and Peterson, B. G. and Ulrich, J. (2016). "'DEoptim': Differential Evolution in 'R'." R package version 2.2-4, URL `https://cran.r-project.org/package=DEoptim`.

Bates, D. and Maechler M. (2016). ***Matrix**: Sparse and Dense Matrix Classes and Methods.* R package version 1.2-7.1, URL `https://CRAN.R-project.org/package=Matrix`.

Blundell, R. and Bond, S. (1998). "Initial conditions and moment restrictions in dynamic panel data models." *Journal of Econometrics*, **87**, 115–143.

Bond, S. and Soderbom, M. (2005). "Adjustment Costs and the Identification of Cobb Douglas Production Functions." *Working Paper.*

Croissant, Y. and Millo, G. (2008) "Panel data econometrics in R: The plm package." *Journal of Statistical Software*, **27**(2), 1–43.

Ghalanos A. and Theussl S. (2015). ***Rsolnp**: General Non–Linear Optimization using Augmented Lagrange Multiplier Method.* R package version 1.16, URL `https://cran.r-project.org/package=Rsolnp`.

Hopenayn, H. A. (1992). “Entry, Exit and Firm Dynamics in Long Run Equilibrium.” *Econometrica*, **60**(5), 1127–1150.

Kleiber, C. and Zeileis, A. (2008). “Applied Econometrics with R.” *Springer-Verlag*, New York.

Levinsohn, J. and Petrin, A. (2003). “Estimating Production Functions Using Inputs to Control for Unobservables.” *The Review of Economic Studies*, **70**(2), 317–341.

Melitz, M. J. (2003). “The Impact of Trade on Intra-Industry Reallocations and Aggregate Industry Productivity.” *Econometrica*, **71**(6), 1695–1725.

Mollisi, V. and Rovigatti, G. (2018). “Theory and Practice of TFP Estimation: the Control Function Approach Using Stata.” *The Stata Journal*, forthcoming.

Mullen, K. and Ardia, D. and Gil, D. and Windower, D. and Cline, J. “DEoptim: An R Package for Global Optimization by Differential Evolution.” *Journal of Statistical Software*, **40**(6).

Olley, S. G. and Pakes, A. (1996). “The Dynamics of Productivity in the Telecommunications Equipment Industry.” *Econometrica*, **64**(6), 1263–1297.

Price, K. V. and Storn, R. M. and Lampinen, J. A. (2006). “Differential Evolution - A Practical Approach to Global Optimization.” *Berlin Heidelberg: Springer-Verlag*, ISBN 3540209506.

Robinson, P. M. (1988). “Root-N-consistent semiparametric regression.” *Econometrica: Journal of the Econometric Society*, 931–954.

Rovigatti, G. (2017). “Production Function Estimation in R: The **prodest** package.” R package version 0.1.1

R Core Team (2017). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL `https://www.R-project.org/`.

Schmidberger, M. and Morgan, M. and Eddelbuettel, D. and Yu, H. and Tierney, L. and Mansmann, U. “State-of-the-art in Parallel Computing with R.” *Journal of Statistical Software*, **47**(1).

Ye, Y. (1988). “Interior algorithms for linear, quadratic, and linearly constrained non linear programming.” *Stanford University, PhD Thesis.*

Wickham, H. (2011). “The Split-Apply-Combine Strategy for Data Analysis.” *Journal of Statistical Software*, **40**(1), 1–29. URL `http://www.jstatsoft.org/v40/i01/`.

Wooldridge, J. M. (1996). “Estimating systems of equations with different instruments for different equations.” *Journal of Econometrics*, **74**, 387–405.

Wooldridge, J. M. (2001). “Applications of generalized method of moments estimation.” *The Journal of Economic Perspectives*, **15**(4), 87–100.

Wooldridge, J. M. (2002). “Econometric analysis of cross section and panel data.” *MIT press.*

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

Affiliation:

Gabriele Rovigatti
Bocconi University
Via Roberto Sarfatti, 25
20100, Milano, IT
E-mail: gabriele.rovigatti@gmail.com
URL: https://sites.google.com/view/gabrielerovigatti/home