1

Introduction

This book is about doing panel data econometrics with the R software. As such, it is aimed at both panel data analysts who want to use R and R users who endeavor in panel data analysis. In this introductory chapter, we will motivate panel data methods through a simple example, performing calculations in base R, to introduce panel data issues to the R user; then we will give an overview of econometric computing in R for the analyst coming from different software packages or environments.

1.1 Panel Data Econometrics: A Gentle Introduction

In this section we will introduce the broad subject of panel data econometrics through its features and advantages over pure cross-sectional or time-series methods. According to Baltagi (2013), panel data allow to control for individual heterogeneity, exploit greater variability for more efficient estimation, study adjustment dynamics, identify effects one could not detect from cross-section data, improve measurement accuracy (micro-data instead of aggregated), use one dimension to infer about the other (as in panel time series).

From a statistical modeling viewpoint, first and foremost, panel data techniques address one broad issue: unobserved heterogeneity, aiming at *controlling for unobserved variables possibly biasing estimation*.

Consider the regression model

$$y = \alpha_o + \beta_o x + \gamma_o z + \epsilon_o$$

where x is an observable regressor and z is unobservable. The feasible model on observables

$$y = \alpha + \beta x + \epsilon$$

suffers from an omitted variables problem; the **OLS** estimate of $\hat{\epsilon}$ is consistent if z is uncorrelated with either y or x: otherwise it will be biased and inconsistent.

One of the best-known examples of unobserved individual heterogenetiy is the agricultural production function by Mundlak (1961) (see also Arellano, 2003, p. 9) where output y depends on x (labor), z (soil quality) and a stochastic disturbance term (rainfall) so that the data-generating process can be represented by the above model; if soil quality z is known to the farmer, although unobservable to the econometrician, it will be correlated with the effort x and hence $\hat{\beta}_{\mathbf{OLS}}$ will be an inconsistent estimator for β.

This is usually modeled with the general form:

$$y_{nt} = \alpha + \beta^\top x_{nt} + (\eta_n + \nu_{nt}) \tag{1.1}$$

Panel Data Econometrics with R, First Edition. Yves Croissant and Giovanni Millo.
© 2018 John Wiley & Sons Ltd. Published 2018 by John Wiley & Sons Ltd.
Companion website: www.wiley.com/go/croissant/data-econometrics-with-R

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 2

where η_n is a time-invariant, generally unobservable characteristic. In the following we will motivate the use of panel data in the light of the need to control for unobserved heterogeneity. We will eliminate the individual effects through some simple techniques. As will be clear from the following chapters, subject to further assumptions on the nature of the heterogeneity there are more sophisticated ways to control for it; but for now we will stay on the safe side, depending only on the assumption of time invariance.

1.1.1 Eliminating Unobserved Components

Panel data turn out especially useful if the unobserved heterogeneity z is (can be assumed) time-invariant. Leveraging the information on time variation for each unit in the cross section, it is possible to rewrite the model (1.1) in terms of observables only, in a form that is equivalent as far as estimating β is concerned. The simplest one is by subtracting one cross section from the other.

1.1.1.1 Differencing Methods

Time-invariant individual components can be removed by first-differencing the data: lagging the model and subtracting, the time-invariant components (the intercept and the individual error component) are eliminated, and the model

$$\Delta y_{nt} = \beta^\top \Delta x_{nt} + \Delta \nu_{nt} \tag{1.2}$$

(where $\Delta y_{nt} = y_{nt} - y_{n,t-1}$, $\Delta x_{nt} = x_{nt} - x_{nt-1}$ and, from (1.1), $\Delta \epsilon_{nt} = \epsilon_{nt} - \epsilon_{n,t-1} = \Delta \epsilon_{nt}$ for $t = 2, \dots, T$) can be consistently estimated by pooled **OLS**. This is called the *first-difference*, or **FD** estimator.

1.1.1.2 LSDV Methods

Another possibility to account for time-invariant individual components is to explicitly introduce them into the model specification, in the form of individual intercepts. The second dimension of panel data (here: time) allows in fact to estimate the η_ns as further parameters, together with the parameters of interest β. This estimator is referred to as *least squares dummy variables*, or **LSDV**. It must be noted that the degrees of freedom for the estimation do now reduce to $NT - N - K$ because of the extra parameters. Moreover, while the $\hat{\beta}$ vector is estimated using the variability of the full sample and therefore the estimator is NT-consistent, the estimates of the individual intercepts $\hat{\eta}_n$ are T-consistent, as relying only on the time dimension. Nevertheless, it is seldom of interest to estimate the individual intercepts.

1.1.1.3 Fixed Effects Methods

The **LSDV** estimator is adding a potentially large number of covariates to the basic specification of interest and can be numerically very inefficient. A more compact and statistically equivalent way of obtaining the same estimator entails transforming the data by subtracting the average over time (individual) to every variable. This, which has become the standard way of estimating fixed effects models with individual (time) effects, is usually termed *time-demeaning* and is defined as:

$$y_{nt} - \bar{y}_{n.} = (x_{nt} - \bar{x}_{n.})\beta + (\nu_{nt} - \bar{\nu}_{n.}) \tag{1.3}$$

where $\bar{y}_{n.}$ and $\bar{x}_{n.}$ denote individual means of y and X.

This is equivalent to estimating the model

$$y_{nt} = \alpha_n + x_{nt}\beta + \nu_{nt},$$

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 3

i.e., leaving the individual intercepts free to vary, and considering them as parameters to be estimated. The estimates $\hat{\alpha}_n$ can subsequently be recovered from the **OLS** estimation of time-demeaned data.

Example 1.1 individual heterogeneity – `Fatalities` data set
The Fatalities dataset from Stock and Watson (2007) is a good example of the importance of individual heterogeneity and time effects in a panel setting.

The research question is whether taxing alcoholics can reduce the road's death toll. The basic specification relates the road fatality rate to the tax rate on beer in a classical regression setting:

$$frate_n = \alpha + \beta beertax_i + \epsilon_n.$$

Data are 1982 to 1988 for each of the continental US states.

The basic elements of any estimation command in R are a `formula` specifying the model design and a dataset, usually in the form of a `data.frame`. Pre-packaged example datasets are the most hassle-free way of importing data, as needing only to be called by name for retrieval. In the following, the model is specified in its simplest form, a bivariate relation between the death rate and the beer tax.

```
data("Fatalities", package="AER")
Fatalities$frate <- with(Fatalities, fatal / pop * 10000)
fm <- frate ~ beertax
```

The most basic step is a cross-sectional analysis for one single year (here, 1982). One proceeds first creating a model object through a call to `lm`, then displaying a `summary.lm` of it. Printing to screen occurs when interactively calling an object by name. Notice that subsetting can be done inside the call to `lm` by feeding an expression that solves into a logical vector to the `subset` argument: data points corresponding to `TRUE`s will be selected, `FALSE`s discarded.

```
mod82 <- lm(fm, Fatalities, subset = year == 1982)
summary(mod82)

Call:
lm(formula = fm, data = Fatalities, subset = year == 1982)

Residuals:
   Min     1Q Median     3Q    Max
-0.936 -0.448 -0.107  0.230  2.172

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.010      0.139   14.46   <2e-16 ***
beertax        0.148      0.188    0.79     0.43
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.67 on 46 degrees of freedom
Multiple R-squared:  0.0133,  Adjusted R-squared:  -0.00813
F-statistic: 0.621 on 1 and 46 DF,  p-value: 0.435
```

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 4

The beer tax turns out statistically insignificant. Turning to the last year in the sample (and employing `coeftest` for compactness):

```
mod88 <- update(mod82, subset = year == 1988)
library("lmtest")

coeftest(mod88)

t test of coefficients:

            Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.859      0.106   17.54   <2e-16 ***
beertax        0.439      0.164    2.67    0.011 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

the coefficient is significant *and positive*! Similar results appear for any single year in the sample.

Pooling all cross sections together, without considering any form of individual effect, can be done using the regular `lm` function or, equivalently, `plm`; in this second case, for reasons which will be clearer in the following, this is not the default behavior, so the optional `model` argument has to be specified, setting it to `'pooling'`.

Drawing on this much enlarged dataset does not change the result:

```
library("plm")
poolmod <- plm(fm, Fatalities, model="pooling")
coeftest(poolmod)

t test of coefficients:

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.8533     0.0436   42.54  < 2e-16 ***
beertax       0.3646     0.0622    5.86  1.1e-08 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Taxing beer would seem to increase the number of deaths from road accidents so that, extending this line of reasoning far beyond what the given evidence supports, i.e., far outside the given sample, one could even argue that free beer might lead to safer driving. Similar results, contradicting the most basic intuition, appear for any single year in the sample.

Panel data analysis will provide a solution to the puzzle. In fact, we suspect the presence of unobserved heterogeneity: in specification terms, we suspect the restriction $\alpha_n = \alpha \forall n$ in the more general model

$$frate_{nt} = \alpha_n + \beta beertax_{nt} + \epsilon_{nt}$$

to be invalid. If omitted from the specification, the individual intercepts – but for a general mean – will end up in the error term; if they are not independent of the regressor (here,

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 5

if unobserved state-level characteristics are related to how the local beer tax is set) the **OLS** estimate will be biased and inconsistent.

As outlined above, the simplest way to get rid of the individual intercepts is to estimate the model in differences. In this case, we consider differences between the first and last years in the sample. A limited amount of work on the dataset would be sufficient to define a new variable $\Delta_5 y_{nt} = y_{nt} - y_{n,t-5}$ but, as it turns out, for reasons that will become clear in the following chapters, the `diff` method well-known from time series does work in the correct way when applied to panel data through the **plm** package, i.e., `diff(y, s)` is correctly calculated as $y_{nt} - y_{nt-s}$:

```
dmod <- plm(diff(frate, 5) ~ diff(beertax, 5), Fatalities, model="pooling")
coef(dmod)
     (Intercept) diff(beertax, 5)
        -0.02524         -0.95554
```

Estimation on five-year differences finally yields a sensible result: after controlling for state heterogeneity, higher taxation on beer is associated with a lower number of fatalities.

As discussed, another way to control for time-invariant unobservables is to estimate them out explicitly. Separate intercepts could be easily added in plain R using the formula syntax:

```
lsdv.fm <- update(fm, . ~ . + state - 1)
lsdvmod <- lm(lsdv.fm, Fatalities)
coef(lsdvmod)[1]
beertax
-0.6559
```

The estimate is numerically different but supports the same qualitative conclusions.

Fixed effects (*within*) estimation yields an equivalent result in a more compact and efficient way. Specifying `model='within'` in the call to `plm` is not necessary because this estimation method is the default one.

```
library("plm")
femod <- plm(fm, Fatalities)
coeftest(femod)

t test of coefficients:

        Estimate Std. Error t value Pr(>|t|)
beertax   -0.656      0.188   -3.49  0.00056 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The fixed effects model, requiring only minimal assumptions on the nature of heterogeneity, is one of the simplest and most robust specifications in panel data econometrics and often the benchmark against which more sophisticated, and possibly efficient, ones are compared and judged in applied practice. Therefore it is also the default choice in the basic estimating function `plm`.

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 6

Example 1.2 no heterogeneity – `Tileries` data set
There are cases when unobserved heterogeneity is not an issue. The `Tileries` dataset contains data on output and labor and capital inputs for 25 tileries in two regions of Egypt, observed over 12 to 22 years. We estimate a production function. The individual units are rather homogeneous, and the technology is standard; hence, most of the variation in output is explained by the observed inputs. Here, a pooling specification and a fixed effects one give very similar results, especially if restricting the sample to one of the two regions considered:

```
data("Tileries", package = "pder")
coef(summary(plm(log(output) ~ log(labor) + machine, data = Tileries,
             subset = area == "fayoum")))
            Estimate Std. Error   t-value  Pr(>|t|)
log(labor) 0.9174031    0.04661 19.681312 2.933e-45
machine    0.0001074    0.01244  0.008638 9.931e-01
```

```
coef(summary(plm(log(output) ~ log(labor) + machine, data = Tileries,
             model = "pooling", subset = area == "fayoum")))
            Estimate Std. Error t-value  Pr(>|t|)
(Intercept) 0.173423    0.07054  2.4584 1.493e-02
log(labor)  0.964845    0.03818 25.2705 3.992e-60
machine     0.002243    0.01000  0.2242 8.228e-01
```

Notice that we have employed yet another way of compactly looking at the coefficients' table only, instead of printing the whole model summary: the `coef.plm` extractor method, applied to a `summary.plm` object.

By the object orientation of R, applying `coef` to a *model* or to the *summary of a model* – in object terms, to a `plm` or to a `summary.plm` – will yield different results. The curious reader might want to try it himself.

In the following chapters we will see how to test formally for the absence of significant individual effects. For now let us concentrate on how to get things done in R, and the relation to how you would in some other environments.

1.2 R for Econometric Computing

R is widely considered a powerful tool with a relatively steep learning curve. This is true only up to a point as far as econometric computing with R is considered. In fact, rather than complicated, R is *scalable*: it can adapt to the level of difficulty/proficiency adequate for the current user. One might say that R is a "complicated" statistical tool in the same way as a drill is a more complicated tool than a hammer, or a screwdriver. Just like a drill, nevertheless, R *can* actually turn screws: although it can also do so much more.[1]

In a sense, R encompasses most other econometric software, with the exception of that based exclusively on a graphical user interface. While the effective way to use R for econometric computing is to take advantage from its peculiarities, e.g., leveraging the power of object orientation,

1 A drill can be used in place of a hammer for driving nails too, although with limited efficiency. So can R; but this is another story.

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 7

it is in fact possible to mimic in R both the *modus operandi* of procedural statistical packages and of course the functionality of other matrix languages.

In the following we will briefly hint at effective ways to perform econometric computing in R, referring the reader to Kleiber and Zeileis (2008) for a more complete treatment; then, in order to provide a friendly introduction to users of different software, we will show how R can be employed the way one would use a "canned" statistical package, or a "hard-boiled" matrix language.

1.2.1 The Modus Operandi of R

R can be used interactively, issuing one command at a time and reading the results from the session log; or it can be operated in batch mode, writing and then executing an R script. The two modes usually mix up, in that even if one writes commands in an editor, it is customary to execute them one by one, or possibly in small groups.

An edited `.R` file has a number of advantages, first of all that the whole session will be completely reproducible as long as the original data are available. There are nevertheless ways to recover all statements used from a session log, which can be turned into an executable .R script with a reasonable amount of editing, or even more easily from the command history, so that if one starts loosely performing some exploratory calculation and then changes his or her mind, perhaps because of some interesting result, nothing is lost. In short, after an interactive session, one can save:

- the session log in a text file (`.txt`)
- the command history in a text file (`.Rhistory`)
- the whole workspace, or a selection of objects, in a binary file (`.Rdata` or, respectively, `.rda`)

From a structured session's approach, there are two competing approaches to the preservation of a reproducible statistical analysis, like one that led to writing a scientific paper: either "the data are real,", or "the commands are real." In the first case, one saves all the objects that have been created during the work session: perhaps the original data, as read from the original source into a `data.frame` but most importantly the model, and possibly test, objects produced by the statistical procedures so that each one can be later (re)loaded, inspected, and printed out, yielding the needed scientific results. In the second case, the original data are kept untransformed, next to plain text files containing all the R statements necessary for full reproduction of the given analysis. This can be done by simply conserving the data file and one or more .R files containing the procedures or in more structured formats like the popular Sweave framework and utility (Leisch, 2002) whereby the whole scientific paper is dynamically reproducible.

The "commands are real" approach has the advantage of being entirely based on human-readable files (supposing the original data are also, as is always advisable, kept in human-readable format), and its clarity is hard to surpass. Any analysis is reproducible on every platform where R can be compiled, and any file is open to easy inspection in a text editor, should anything go wrong, while binary files, even from Open Source software like R, are always potentially prone to compatibility problems, however unlikely. But considerations on computational and storage demands also play a role.

Computations are performed just once in the first case but for the (usually inexpensive) extraction of results from already estimated model objects and at each reproduction in the second so that the "real data" approach can be preferable, or even the only practical alternative, for computationally heavy analyses. By contrast, the "real commands" approach is much more parsimonious from the viewpoint of storage space, as besides the original data one only needs to archive some small text files.

UNCORRECTED PROOFS

1.2.2 Data Management

1.2.2.1 Outsourcing to Other Software

In the same spirit, although R is one of the best available tools for managing data, users with only a casual knowledge of it can easily preprocess the data in the software of their choice and then load them into R. The **foreign** package (R Core Team, 2017) provides easy one-step import from a number of popular formats. Gretl (Cottrell and Lucchetti, 2007) took it one step further, providing the ability to call R from inside Gretl and to send to it the current dataset. In general, passing through a conversion into tab- (or space-, or comma-) delimited text and a call to the `read.table` function will solve most import problems and provide an interface between R and anything else, including spreadsheets.

1.2.2.2 Data Management Through Formulae

Even at this level one should notice, however, that R formulae are very powerful tools, accepting a number of transformations that can be done "on the fly" eliminating most of the need for data pre-processing. An obvious example are logs, lags, and differences or, as seen above, the inclusion of dummy variables. Power transformations and interaction terms can also be specified inside formulae in a very compact way. A limited investment of time can let even the casual user discover that most of his usual pre-processing can be disposed of, leaving a clean process from the original raw dataset to the final estimates.

Perhaps the use of formulae in R is the first investment an occasional user might want to do, for all the time and errors it saves by streamlining the flow between the original data and the final result.

1.3 plm for the Casual R User

This book is best for readers with familiarity with the basics of R. Nevertheless, using R interactively – the way econometric software is usually employed – to perform most of the analyses presented here requires very few language-related concepts and only three basic abilities:

- how to import data,
- which commands to issue to obtain estimates,
- optionally, how to save the output to a text file or render it toward LaTeX (but one could as well copy results from the active session).

This corresponds to the typical work flow of a statistician using specialized packages, where one issues one single high-level command, possibly of a very rich nature and with lots of switches, performing some complicated statistical procedure in batch mode, and gets the standard output printed out on screen.

Distinctions are of course sharper than this, and the boundaries between specialized packages, where macro commands perform batch procedures, and matrix languages, where in principle estimators have to be written down by the user, are blurred. In fact, and with time, packages have grown proprietary programming features and sometimes matrix languages of their own, so that much development on the computational frontier of econometric methods can be done by the users in interpreted language, just as happens in the R environment, rather than provided in compiled form by the software house. A notable example of this convergence is Gretl (Cottrell and Lucchetti, 2007), a GUI-based open- source econometric package with full-featured scripting capabilities, entirely programmable and extensible. Some well-known commercial offerings have also taken similar paths.

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 9

From the other end of the spectrum, matrix languages have built up huge libraries of ready-made, high-level functions performing complex procedures in one go.

In the following, for the sake of exposition, we will stick to cliche and assume that users of procedural languages expect to run a regression issuing one single command, although perhaps with a lot of arguments, and obtain a lengthy and very comprehensive output containing all the estimation results and diagnostics they might ever need, while matrix language users seek to perform regressions from scratch as $\hat{\beta} = (X^\top X)^{-1} X^\top y$, and obtain any post-estimation diagnostics in the same fashion.

1.3.1 R for the Matrix Language User

The latter viewpoint in our stylized world is that of die-hard econometricians-programmers, who do anything by coding estimators in matrix language. Understandably, the transition toward R is easier done in this case, as it too is a matrix language in its own right. Armed with some cheat sheet providing the translation of basic operators, users of matrix languages can be up and running in no time, learning the important differences in syntax and the language idiosyncrasies of R along the way. As for the moment, here is how linear regression "from scratch" is done in R:

Example 1.3 linear regressions – `Fatalities` data set
In order to perform linear regression "by hand" (*i.e.*, without resorting to a higher level function than simple matrix operators), we have to prepare the y vector and the X matrix, intercept included and then use them in the R translation of the least squares formula:

```
y <- Fatalities$frate
X <- cbind(1, Fatalities$beertax)
beta.hat <- solve(crossprod(X), crossprod(X,y))
```

Notice the use of the numerically efficient operators `solve` and `crossprod` instead of the plain syntax `solve(t(X) %*% X) %*% t(X) %*% y`, which – up to the numerically worst conditioned cases – would produce identical results. (Notice also that we do not need to explicitly make a vector of ones: binding by column (`cbind`-ing) the scalar 1 to a vector of length N, the former is recycled as needed.)

Next, we check that our hand-made calculation produces the same coefficients as the higher-level function `lm`:[2]

```
beta.hat
       [,1]
[1,] 1.8533
[2,] 0.3646
mod <- lm(frate ~ beertax, Fatalities)
coef(mod)
(Intercept)     beertax
     1.8533      0.3646
```

2 Notice that although the coefficients produced by the two methods are numerically the same, from a software viewpoint they are two different object types: the former a 2×1 `matrix`, the latter a (named) `numeric` vector.

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 10

It is less straightforward to perform an **LSDV** or a fixed effects analysis. In the former case, one must create a matrix of state dummy variables: this is cumbersome to do in plain matrix language but is much easier if leveraging the features of R's formulae: in the latter case, it is enough to add the individual index under form of a `factor`: *i.e.*, the R type for qualitative variables.[3]

```
LSDVmod <- lm(frate ~ beertax + state - 1, Fatalities)
coef(LSDVmod)["beertax"]
beertax
-0.6559
```

Estimation is also relatively easy in the fixed effects case, provided that a peculiar feature of R without an obvious counterpart in other matrix languages steps in: ragged arrays. In the following snippet, the `mean` function is *applied* along the individual index to obtain the time means for each individual, which are then replicated along the length of the time dimension. The vectors of time averages are then subtracted from the original vectors to obtain the time-demeaned data, on which plain **OLS** can be applied (`attach` and `detach` are used to bring the contents of the `data.frame` to user level, to avoid having to point at each variable through the `Fatalities$...` prefix).

```
attach(Fatalities)
frate.tilde <- frate - rep(tapply(frate, state, mean),
                           each = length(unique(year)))
beertax.tilde <- beertax - rep(tapply(beertax, state, mean),
                           each = length(unique(year)))
lm(frate.tilde ~ beertax.tilde - 1)

Call:
lm(formula = frate.tilde ~ beertax.tilde - 1)

Coefficients:
beertax.tilde
       -0.656
detach(Fatalities)
```

This simple example already gives an idea of the small computational complications arising from **LSDV** or fixed effects estimation. For example, it would not work for unbalanced panels *as is*. The simple modification required to generalize the above snippet to the unbalanced case is left as an exercise for the willing reader.

1.3.2 R for the User of Econometric Packages

The opposite vision is to resort to macro commands. At a bare minimum, users who are familiar with procedural languages can obtain the same result with R:

- issue estimation command,
- get printed output

3 Text labels like state names would be automatically converted, while numerical codes would not. In the latter case, one would use `as.factor(state)` within the `formula`.

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 11

despite the logical separation between the steps of creating a model object, summarizing, it and printing the summary, which can a) be executed separately but can also b) be nested inside the same statement, exploiting the functional logic of R, by which "inner" arguments are evaluated first, (implicitly) printing the summary of a model object which is estimated on the fly inside the same statement.[4] Easier done than said:

```
summary(plm(fm, Fatalities))
Oneway (individual) effect Within Model

Call:
plm(formula = fm, data = Fatalities)

Balanced Panel: n = 48, T = 7, N = 336

Residuals:
    Min.  1st Qu.   Median  3rd Qu.     Max.
-0.58696 -0.08284 -0.00127  0.07955  0.89780

Coefficients:
        Estimate Std. Error t-value Pr(>|t|)
beertax   -0.656      0.188   -3.49  0.00056 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:    10.8
Residual Sum of Squares: 10.3
R-Squared:      0.0407
Adj. R-Squared: -0.12
F-statistic: 12.1904 on 1 and 287 DF, p-value: 0.000556
```

The construct `summary(myestimator(myformula, mydata, ...))` will generally work, displaying estimation results to screen, for most estimators. Diagnostics will often have a `formula` method so that a statement along the lines of `mytest(myformula, mydata, ...)` will produce the desired output, or, at most, they will require the trivial task of making a "model" object before applying the desired test to it: which can as well happen in one single statement, like `mytest(myestimator(myformula, mydata, ...))`. In this sense, R is a good substitute of procedural languages, at least those that require text input from the command line; despite the fact of also being so much more.

If one is not scared of typing, we might even say that inputting the above statement is not far from the level of difficulty of using a point-and-click **GUI**. Sure it is not any more difficult to read output from the above R command than that of the standard regression in a **GUI** package.

1.4 plm for the Proficient R User

A better knowledge of R will disclose a wealth of possibilities streamlining the production process of empirical research. Actually, while R might look difficult or unfriendly to the beginner,

4 Intentionally convoluted sentence. This is what actually happens under the bonnet, but the user need not necessarily worry about it.

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 12

for the proficient user the overall workload when producing a piece of scientific research may turn out to be much lower than with competing solutions. The convenient features that allow for a more advanced management of research activity with respect to the usual paradigm "analyze the data – save the results – write the paper around them" can also be seen in the light of producing reproducible econometric research.

1.4.1 Reproducible Econometric Work

Performing econometric work in R, possibly in conjunction with LaTeX through literate statistical tools like Sweave (Leisch, 2002) and knitr (Xie, 2015), satisfies desirable standards of reproducibility.

Following Peng (2011), "[an] important barrier [to reproducible research] is the lack of an integrated infrastructure for distributing [it] to others." Yet such infrastructures have recently emerged in statistics and have been proposed for econometric practice. As advocated by Koenker and Zeileis (2009), one way of ensuring the complete reproducibility of one's research is to provide a self-contained Sweave file – "a tightly coupled bundle of code and documentation" – including all the text as well as the code generating the results of the paper so that, given the original data, the complete document can be reproduced exactly by anybody, on practically any computing platform.

Three aspects of R are worth highlighting in this context: object orientation; code availability, documentation, and management; and reproducible econometric research through literate programming functionalities. The latter two, in particular, help situate econometric work (properly) done with R toward the better end of the reproducibility spectrum in Peng (2011), the "gold standard" of full replication, as providing "a detailed log of every action taken by the computer," which can be replicated by anyone with any type of machine and an Internet connection. In this sense, R code is *linked* and *executable* (Peng, 2011, Fig.1) without the need for either proprietary software or particular hardware/operating system, with the only possible limit of computing power.

As for *availability*, R is open-source software (OSS); hence, all code can be used, inspected, copied, and possibly modified at will. Source code, in the words of Koenker and Zeileis (2009), is "the ultimate form of documentation for computational science," and being accessible it can more easily be subjected to critical scrutiny (on the subject, see also Yalta and Lucchetti, 2008; Yalta and Yalta, 2010).

Besides accessibility, being OSS has important consequences on numerical accuracy (see Yalta and Yalta, 2007) and, what matters most here, on the particular aspect of reproducibility. The R project encourages (in a sense, enforces) documentation of code through its packaging system: in order for a package to build, every (user-level) function inside it must be properly documented, with valid syntax and working examples, as checked by automated scripts. Reliability levels are explicit too: the main distribution site, the Comprehensive R Archive Network (cran.r-project.org) accepts stable versions of packages, subject to a further validation step; earlier versions of code, labeled according to development status (from "Planning" to "Mature"), are to be found on collaborative development platforms of which R-Forge (r-forge.r-project.org/) (Theußl and Zeileis, 2009) is a prominent example. The latter, although typically containing very recent methods, are subject to all the above mentioned quality controls but also allow for immediate patching of code; all changes are tracked inside the system's version history and are open to inspection from any user.

Lastly, and perhaps most importantly here, R explicitly encourages reproducibility of research through utilities like Sweave (Leisch, 2002), which implements literate programming techniques weaving together code and documentation in a dynamic document, as discussed

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 13

in Meredith and Racine (2009) and Koenker and Zeileis (2009, 2.5). Convenient interfaces for weaving together R and LaTeX are available, from Emacs + ESS (Rossini et al., 2004) to the more recent RStudio (Racine, 2012). This book has in fact been prepared as a dynamic LaTeX document, using the Emacs editor in ESS mode.

1.4.2 Object-orientation for the User

R has object-orientation features. Beside their user-friendliness, such features have a role of their own in reproducibility: simplifying the code makes it more readable and using modular, high-level components with sensible defaults for the different objects is generally safer, especially for the accident-prone data manipulations and transformations typical of panel data.

Methods for extracting (individual, average, or pooled) coefficients, standard errors and measures of fit from model objects of different kinds work with the same syntax, although with different internals, transparently for the user. Formulae with compact representations of lags and differences can be supplied to panel estimators, where the above operators will automatically adjust to the particular context of panel data. Moreover, compact formulations of dynamic models can be indexed, as in `lag(x, 1:i)` for $x_{t-1}, \ldots, x_{t-i}$, and used inside flow control structures, simplifying the making of large tables. Preliminary data manipulation can often be avoided altogether, calculating lags, differences, logs, or more specific panel operations, such as averaging or demeaning over the time or individual dimension, inside the model formula. As observed before, this generally allows to maintain only two files: the original data source and the procedures, with obvious benefits to reliability and replicability of results.

The flexibility object-orientation features provide is highlighted when considering that the R workspace can contain objects of many different kinds at the same time: in this instance, panel or simple models, model formulae, matrices or lists of weights for representing spatial dependence, and, differently from some widespread econometric packages, datasets of various dimensions at the same time. Such flexibility is particularly useful in research work that blends methods from different lines of research together, in order to avoid having to use different software environments for the tasks at hand, and the common pitfalls of not saving the code relative to preliminary data manipulations, or that which combines the results together (see Peng, 2011, p. 1226).

1.5 plm for the R Developer

The last frontier for **plm** users is to become developers. The operation of **plm** is based on a specific data infrastructure able to deal with the peculiar aspects of panel data: basically, their double indexing feature, the possibility of *unbalancedness*, and the frequent need for transformations along one (or both) dimension(s). This mid-level functionality for (panel) data transformation is in general accessible at user level and can be very handy for those developing new methods, e.g., involving estimation over transformed data. It is in fact already in use by a number of other packages: in particular, but not only, some packages aimed at more specific needs presented in this book (**pglm**, **splm**), which are based on this infrastructure and are mostly compliant with **plm**'s conventions and syntax.

Just as the econometric estimation of a fixed effects model proceeds through applying standard OLS to demeaned data, so does the implementation in **plm**, like many others. Yet, unlike many other software packages, here these steps can be readily performed in an explicit fashion.

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 14

Example 1.4 explicit *within* transformation – `Fatalities` data set
In order to demonstrate *within* regression, we apply the transformation functions directly in the model formula, excluding *a priori* the intercept (which has been transformed out):

```
w.mod <- plm(Within(frate) ~ Within(beertax) - 1, data=Fatalities,
             model = "pooling")
coef(w.mod)
Within(beertax)
        -0.6559
```

(If trying this at home, remember that, unlike the coefficient, the standard error from this model's output would have to be adjusted by the degrees of freedom to match that of the canned *within* routine. See the discussion in the next chapter, 2.2.3.)

As often happens with R, "ideas are turned into software" (Chambers, 1998) in a natural way, the computational approach following the conceptual flow of the statistical reasoning. Moreover, while all of the software tools provided, being open-source, can ultimately be inspected by the skilled programmer, in the case of **plm** much of the infrastructure is available at user level, conveniently packaged with help and examples, both for instruction purposes and as a building block for further development.

1.5.1 Object Orientation for Development

One last observation is in order, whose scope is not limited to **plm** or panel data econometrics. For a developer, working inside the R project has the huge benefit that she is able to access a majority of all available statistical techniques from *inside* her preferred computing environment, by simply loading the relevant package. In our particular field, this means that one can leverage functionality from, say, general statistics, such as, e.g., using principal components analysis to approximate common factors (see Chapter 9); or from quantitative geography, such as calculating distances between the centroids of regions to make spatial weights matrices (see Chapter 10). This has to do with the *functional* orientation of R, by which complex (statistical) tasks are *abstracted* into functions and therefore made available irrespective of the internals (what happens under the hood).

Another side of abstraction is *object orientation*: generic methods are often provided, which particularize into different actual computations depending on the object they are fed. Simple examples are `summary` and `plot`, which will produce different outcomes if applied to, say, a `numeric` or an `lm`.

A related, relevant feature of R, and in general of the S language (Chambers, 1998), for the developer is that *functions are a data type*. This means that a function (the abstraction of a statistical procedure) can be passed on to another statistical procedure simply calling it by name. A simple example is the case of the Wald test for generic linear restrictions of the form $R\gamma = r$ on the parameter vector γ:

$$\text{Wald}(R\hat{\gamma} - r)^\top [R^\top \hat{\text{V}} R]^{-1} (R\hat{\gamma} - r) \tag{1.4}$$

Taking the **OLS** estimate of the linear model as an example, the standard – or "classical" – covariance matrix $\hat{\text{V}}_{\textbf{OLS}} = \frac{\sum_{n=1}^{N} \hat{\epsilon}^2}{N} (Z^\top Z)^{-1}$ will only be appropriate if the errors are independent and identically distributed. If heteroscedasticity is present, the parameter

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 15

estimates $\hat{\gamma}_{\text{OLS}}$ are still consistent, but $\hat{\text{V}}_{\text{OLS}}$ is not. The test can then be robustified employing a heteroscedasticity-consistent covariance estimator in place of V_{OLS} (Zeileis, 2006a).

The R counterpart of the Wald test is the `linearHypothesis` function, aliased by the abbreviation `lht` (Fox and Weisberg, 2011). Mimicking the relevant statistical procedure, the latter will use `coef` and – by default – `vcov` methods to extract $\hat{\gamma}$ and $\hat{\text{V}}$ from the estimated model, plugging them into (1.4). By default, an `lm` object will contain $\hat{\text{V}}_{\text{OLS}}$ but the user can, optionally, provide *a different way to calculate the covariance* under form of the function argument `vcov`.

Example 1.5 Wald test with user-supplied covariance – `Tileries` data set
As previously seen, the production function model in the `Tileries` dataset is a good candidate for a pooling specification. Below, for the sake of exposition, we estimate a linearized Cobb-Douglas version of the production function, in order to test a hypothesis of constant returns to scale. It seems appropriate, as a first step, to estimate a pooled specification by **OLS**:

```
data("Tileries", package = "pder")
til.fm <- log(output) ~ log(labor) + log(machine)
lm.mod <- lm(til.fm, data = Tileries, subset = area == "fayoum")
```

before proceeding to test the restriction $H_0 : \gamma_1 + \gamma_2 = 1$

```
library(car)
lht(lm.mod, "log(labor) + log(machine) = 1")
Linear hypothesis test

Hypothesis:
log(labor)  + log(machine) = 1

Model 1: restricted model
Model 2: log(output) ~ log(labor) + log(machine)

  Res.Df   RSS Df Sum of Sq   F Pr(>F)
1    175 0.602
2    174 0.600  1   0.00104 0.3   0.58
```

Allowing for heteroscedasticity is as easy as passing on `vcovHC` to the `vcov` argument:

```
library(car)
lht(lm.mod, "log(labor) + log(machine) = 1", vcov=vcovHC)
Linear hypothesis test

Hypothesis:
log(labor)  + log(machine) = 1

Model 1: restricted model
Model 2: log(output) ~ log(labor) + log(machine)

Note: Coefficient covariance matrix supplied.

  Res.Df Df    F Pr(>F)
1    175
2    174  1 0.23   0.63
```

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 16

The qualitative findings are unchanged, but this is not the point. As the `Note` in the output reminds us, a different covariance estimator has been employed.

Being generic methods, both `lht` and `vcovHC` will select and apply the appropriate particular procedure depending on the object type. Thus, if fed an `lm`, inside `lht.lm` `coef.lm` and `vcovHC.lm` will be applied, with the relevant defaults; if a `plm` is provided instead, `coef.plm` and `vcovHC.plm` will be used.

By default, the most appropriate method for estimating the parameters' covariance in a panel setting is by allowing for clustering. This is what will happen if feeding the `vcovHC` function to the `lht` together with a `plm` object: the `vcovHC` generic will select the `vcovHC.plm` method for doing the actual computing.

Example 1.6 user-supplied covariance, continued – `Tileries` data set
The pooled specification by **OLS** can be estimated through `plm` as well:

```
plm.mod <- plm(til.fm, data = Tileries, subset = area == "fayoum")
```

before proceeding to test H_0:

```
library(car)
lht(plm.mod, "log(labor) + log(machine) = 1", vcov = vcovHC)
Linear hypothesis test

Hypothesis:
log(labor)  + log(machine) = 1

Model 1: restricted model
Model 2: log(output) ~ log(labor) + log(machine)

Note: Coefficient covariance matrix supplied.

  Res.Df Df Chisq Pr(>Chisq)
1    167
2    166  1  0.76       0.38
```

Another different covariance has been employed this time, which allows for *clustering* at individual level: an idea that will be explored in Chapter 5. For now it will be sufficient to say that this one, next to heteroscedasticity, allows for error correlation in time within each individual.

Again, constant returns to scale are not rejected; but now our conclusion is valid in a much more general context.

The programmer writing a `lht` method for, say, a hypothetical `mymodel` class will not have to bother about these downstream details because all he needs is for `mymodel` objects to expose `vcov` and `coef` methods and, eventually, to provide alternative covariance estimators, embodied in turn into `vcovXX.mymodel` functions. Then his function will automatically reproduce equation (1.4) in the new context. The **plm** package has been designed to be compliant with this framework and to allow for easy extensions along the lines sketched above.

Next to the issue of designing modular code for easier production and maintenance by re-employing existing functionality in new contexts, object orientation also has important

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 17

computational advantages in terms of efficiency. As we have seen, object orientation means that the statistical "objects" (the coefficient vector, the covariance) are mapped to computational tools according to types. From the point of view of the developer faced with computational efficiency and accuracy issues, this means that often she is able to exploit the peculiar structure of the problem at hand. Specialized methods (usually written and compiled in C or FORTRAN) are often available, speeding up computations by many orders of magnitude for a specific class of problems.

One simple example is the inversion of block-diagonal symmetric matrices; a typical problem in panel data estimation by **gls**, where the estimated error covariance matrix, which is $NT \times NT$, has to be `solve`d. An obvious improvement is to exploit the property that the inverse of a block-diagonal matrix is made of the inverses of the individual blocks; nevertheless, defining the error covariance as a `bdsmatrix` object allows to use the fast `solve.bdsmatrix` method from the package by the same name (Therneau, 2014). This solution is used, e.g., for the **ggls** estimators described in Chapter 5: a procedure for which computational efficiency is critical, as being statistically appropriate for very large N panels, where on the other hand it becomes computationally problematic.

Another instance where special matrix types greatly extend the feasibility boundaries is in spatial models: here, *sparse* matrices are common, which contain a vast majority of zeros. Simplifying, one could say that sparse matrix algebra methods rely on the additional information on the position of zeros, avoiding either to consume memory for storing them, or to waste resources to compute on them. Sparse matrix methods from the package **spam** (Furrer and Sain, 2010) and from the more general matrix algebra package **Matrix** (Bates and Maechler, 2016) have been extensively employed in the spatial panel methods described in Chapter 10, together with optimizers from **nlme** (Pinheiro et al., 2017) and **MaxLik** (Henningsen and Toomet, 2011) (a discussion is to be found in Millo, 2014, Section 5.2).

On a different but related note, innovation in object types has in turn affected the symbolic descriptions of models: *formulae*, from which model matrices and responses are derived for actual computation. The extension of the `formula` object class into the `Formula` class, which inherits from the former generalizing it to allow multi-part models and multiple responses (Zeileis and Croissant, 2010), is the basis for the consistent specification of a number of estimators based on combining different levels of instrumentation. The consistent and flexible **plm** implementation of the econometric methods described in Chapters 6 and 7 is made possible by the extended functionality of `Formulae`.

This book is on using, rather than developing, panel data methods in R. This short discussion, therefore, cannot but scratch the surface of the wealth of computing infrastructure available to the user who turns toward developing her own methods. We hope to have at least given an intuition and some directions for further inquiry to any user of **plm** and related packages who wants to extend the methods contained herein, leveraging the power of the R environment at large. As Borges put it, "This plan is so vast that each writer's contribution is infinitesimal."

1.6 Notations

This book is necessarily notation-heavy. Moreover, conventions differ across the various subfields of panel data econometrics covered herein. A considerable effort has been made to present formalizations in a consistent way across chapters, although sometimes this can entail a departure from the usual habits.

This section is therefore meant as a reference for the whole book.

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 18

1.6.1 General Notation

The probability is denoted by $\boxed{\mathrm{P}}$, the expected value is denoted by $\boxed{\mathrm{E}}$, the variance by $\boxed{\mathrm{V}}$, the trace by $\boxed{\mathrm{tr}}$, the correlation coefficient by $\boxed{\mathrm{cor}}$, and the standard deviation by $\boxed{\sigma}$. A quadratic form is denoted by $\boxed{q}$ and the identity matrix by $\boxed{\mathrm{I}}$. A set of covariates define two matrices, $\boxed{\mathrm{P}}$, which returns the fitted values when post-multiplied by a vector and $\boxed{\mathrm{M}}$, which returns the residuals: $\mathrm{P} = X(X^\top X)^{-1}X$ and $\mathrm{M} = \mathrm{I} - \mathrm{P}$. The Cholesky decomposition of a matrix is denoted by $\boxed{C}$, so that:

$$CAC^\top = \mathrm{I}$$

1.6.2 Maximum Likelihood Notations

For models estimated by the maximum likelihood method, the objective function is denoted by $\boxed{\ln L}$, the Jacobian by $\boxed{J}$, the gradient by $\boxed{g}$, the Hessian by $\boxed{H}$ and the information matrix by I. For generic presentations of the log-likelihood method, the generic set of parameters is denoted by $\boxed{\theta}$.

The statistics of the three tests are denoted by $\boxed{\mathrm{LR}}$, $\boxed{\mathrm{LM}}$, and $\boxed{\mathrm{Wald}}$ for, respectively, a likelihood ratio, a Lagrange multiplier, and a Wald test.

1.6.3 Index

A panel is constituted of $\boxed{N}$ individuals denoted by $\boxed{n}$ (when necessary, $\boxed{m}$ is used as an alias for n).

Each individual is observed during $\boxed{T}$ different periods denoted by t (when necessary, s is used as an alias to the t index).

The size of the sample is denoted by $\boxed{O}$, it is equal to $\sum_{n=1}^{N} T_n$, where T_n is the number of time series for individual n. If $T_n = T\ \forall n$ (balanced panel case), we have $O = NT$.

The $\boxed{K}$ covariates are indexed by $\boxed{k}$; note that a column of ones is not consider in this count.

1.6.4 The Two-way Error Component Model

Consider now the two-way error-component model (the more usual one-way individual error component model is obtained as a special case); it writes for an observation:

$$y_{nt} = \alpha + \beta^\top x_{nt} + \epsilon_{nt} = \gamma^\top z_{nt} + \epsilon_{nt}$$

$$\epsilon_{nt} = \eta_n + \mu_t + \nu_{nt}$$

$\boxed{y}$ is the response, $\boxed{\alpha}$ the intercept, $\boxed{x}$ the vector of K covariates with associated coefficients $\boxed{\beta}$. It would be sometimes easier to consider $\boxed{z}$, which is obtained by adding a 1 in the first position of vector x: $z_{nt}^\top = (1, x_{nt}^\top)$, with the vector of associated coefficient $\boxed{\gamma}$ with $\gamma^\top = (\alpha, \gamma^\top)$.

The error of the model $\boxed{\epsilon}$ is the sum of a time-invariant individual effect $\boxed{\eta}$, an individual-invariant time-effect $\boxed{\mu}$, and a residual error $\boxed{\nu}$. Except for some time-series and spatial methods, ν is assumed to be i.i.d..

The variance is denoted by σ^2; we therefore have for the error and its components, σ_ϵ^2, σ_η^2, σ_μ^2 and σ_ν^2.

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 19

All estimated values are represented by the "true" value with a hat so that, for example, the estimated coefficients, the residuals, and the estimated variance of the errors are respectively: $\hat{\beta}$, $\hat{\epsilon}$, and $\hat{\sigma_\epsilon^2}$.

In matrix form, the same model is written:

$$y = \alpha j + X\beta + \epsilon = Z\gamma + \epsilon$$
$$\epsilon = \mathrm{D}_\eta \eta + \mathrm{D}_\mu \mu + \nu$$

where j is a vector of 1, X and Z the covariate matrices (the latter including a first column of ones, the former without it), η the vector of N individual effects, μ the vector of T time effects, and ν the vector of O residual effects.

D denotes a matrix of dummy variables; D_η and D_μ are respectively the dummy variable matrices for individuals and periods. In the case of balanced panels, and if the observations are ranked first by individual, then by time series ("the t index changes faster"), these two matrices can be expressed using Kronecker products. Denoting by $J = jj^\top$ a square matrix of ones, we have:

$$\mathrm{D}_\eta = \mathrm{I}_N \otimes J_T$$
$$\mathrm{D}_\mu = J_T \otimes \mathrm{I}_N$$

The covariance matrix of the errors ϵ is denoted by Ω. Some simplifying assumptions lead to:

$$\Omega_\epsilon = \sigma_\nu^2 \mathrm{I}_{NT} + \sigma_\eta^2 \mathrm{I}_N \otimes J_T + \sigma_\mu^2 J_T \otimes \mathrm{I}_N$$

or $\Omega_\epsilon = \sigma_\nu^2 \Sigma$, with:

$$\Sigma = \mathrm{I}_{NT} + \frac{\sigma_\eta^2}{\sigma_\nu^2} \mathrm{I}_N \otimes J_T + \frac{\sigma_\mu^2}{\sigma_\nu^2} J_T \otimes \mathrm{I}_N$$

1.6.5 Transformation for the One-way Error Component Model

For the one-way individual error component model, the last term disappears. In this case, we'll denote S the matrix that if post-multiplied by a variable, returns a vector of length O containing the individual sums of the variable, each one being repeated T_N times.

$$S = \mathrm{I}_N \otimes J_T$$

We'll also make use of the matrix $\bar{\mathrm{I}} = \mathrm{I} - \bar{J}$, which post-multiplied by a variable, returns the variable in deviation from its overall mean:

$$\bar{\mathrm{I}} = \mathrm{I} - \bar{J}$$

In the case of balanced panels, the *between* and *within* matrices, respectively denoted by B and W, can be defined:

$$B = \frac{1}{T} S = \frac{1}{T} \mathrm{I}_N \otimes J_T$$
$$W = \mathrm{I} - B = \mathrm{I}_{NT} - \frac{1}{T} \mathrm{I}_N \otimes J_T$$

Denoting $\sigma_\iota^2 = \sigma_\nu^2 + T\sigma_\eta^2$, the covariance matrix of the error can be written:

$$\Omega_\epsilon = \sigma_\nu^2 \left(W + \frac{\sigma_\iota^2}{\sigma_\nu^2} B \right) = \sigma_\nu^2 \left(W + \frac{1}{\phi^2} B \right)$$

UNCORRECTED PROOFS

with $\boxed{\phi} = \frac{\sigma_\nu}{\sigma_i}$. We'll also denote $\boxed{\theta} = 1 - \phi$ the fraction of the individual mean that is subtracted in the **GLS** model.

Two other transformations matrices are used: $\boxed{D}$ and $\boxed{O}$ perform respectively the first-difference and the orthogonal deviation of a vector.

1.6.6 Transformation for the Two-ways Error Component Model

Back to the two-ways error component model, we now have two between matrices:

$$B_\eta = \mathrm{I}_N \otimes J_T/T$$
$$B_\mu = J_T \otimes \mathrm{I}_N/N$$

The *within* matrix is, denoting by $\boxed{\bar{J}} = J_{NT}/NT$ the matrix that post multiplied by a vector representing a variable returns its overall mean repeated NT times:

$$W = \mathrm{I} - B_\eta - B_\mu + \bar{J}$$

Denote further $\bar{B}_\eta = B_\eta - \bar{J}$ and $\bar{B}_\mu = B_\mu - \bar{J}$. The covariance matrix of the errors then writes:

$$\Omega_\epsilon = \sigma_\nu^2 \left(W + \frac{1}{\phi_\eta^2}\bar{B}_\eta + \frac{1}{\phi_\mu^2}\bar{B}_\eta + \frac{1}{\phi_2^2}\bar{J} \right)$$

with:

$$\phi_\eta^2 = \frac{\sigma_\nu}{\sqrt{\sigma_\nu^2 + T\sigma_\eta^2}}$$
$$\phi_\mu^2 = \frac{\sigma_\nu}{\sqrt{\sigma_\nu^2 + N\sigma_\mu^2}}$$
$$\phi_2^2 = \frac{\sigma_\nu}{\sqrt{\sigma_\nu^2 + T\sigma_\eta^2 + N\sigma_\mu^2}}$$

as for the one-way individual error component model, $\theta_i = 1 - \phi_i$ for $i = \eta, \mu, 2$

1.6.7 Groups and Nested Models

The group effect is denoted by $\boxed{\lambda}$, the $\boxed{G}$ groups are indexed by $\boxed{g}$.

1.6.8 Instrumental Variables

The matrix of instruments is denoted by $\boxed{L}$, the number of instruments by $\boxed{M}$.

1.6.9 Systems of Equations

We consider a system of $\boxed{L}$ equations indexed by $\boxed{l}$, aliased by $\boxed{m}$ when necessary. $\boxed{\Xi}$ is the matrix of dimension $O \times L$ where each column contains the error vector for one of the equations. Its covariance matrix is denoted by $\boxed{\Sigma}$:

$$\Sigma = \mathrm{E}(\Xi^\top \Xi)$$

its elements being denoted by $\boxed{\sigma}$.

UNCORRECTED PROOFS

Trim Size: 178mm x 254mm Single Column Croissant c01.tex V1 - 05/02/2018 7:54pm Page 21

1.6.10 Limited Dependent and Count Variables

For the ordered model, the J levels of the dependent variables, defined by the thresholds μ, are denoted by j.

For Poisson models, the Poisson parameter is denoted by θ. In the cross-series case, it is equal to:

$$\theta_n = \eta_n \lambda_{nt}$$

where λ is a linear combination of the covariates. Λ and Y are respectively the sums of λ and of the response for one individual. For the Negbin model, δ is the parameter of the distribution and ν the parameter that links the expected value and the variance of the response.

1.6.11 Time Series

The most general time-series model considered in the book is defined by the following equations:

$$\begin{aligned}
y_{nt} &= \rho y_{t-1} + \gamma^\top z_{nt} + \epsilon_{nt}\\
\epsilon_{nt} &= \eta_n + \nu_{nt}\\
\nu_{nt} &= \psi \nu_{nt-1} + \zeta_{nt}
\end{aligned}$$

where ρ is the auto-regressive coefficient of the AR(1) model, ψ the coefficient of the AR(1) process for ν, and ζ are iid errors.

When lags are needed, they'll be denoted by l and the number of lags is denoted by L. It is sometimes useful to know the beginning of the process; this will be denoted by $-S$, 1 being the first available observation.

1.6.12 Spatial Panels

The proximity matrix is denoted by W, its elements by w. It can be constructed either on binary neighborhood or on the distance d between two individuals. The most general model considered in this book is described by the following equations:

$$\begin{aligned}
y &= \lambda(\mathrm{I}_T \otimes W)y + Z\gamma + \epsilon\\
\epsilon &= (j_T \otimes \eta) + \nu\\
\nu &= \rho(\mathrm{I}_T \otimes W)\nu + \zeta\\
\zeta_t &= \psi\zeta_{t-1} + \xi_t, \quad t = 1, \ldots, T
\end{aligned}$$

The first equation defines a SAR model, and the auto-regressive spatial coefficient for the response is denoted by λ. The third equation defines a SEM model: it indicates that the non-individual part of the error of the model is also spatially auto-correlated, the auto-regressive spatial coefficient being denoted by ρ. Finally, the last equation indicates that the residual component of the model is serially auto-correlated, with an AR(1) coefficient denoted by ψ; and ξ defines i.i.d.errors.

To simplify the notation, we'll also define matrices $A = \mathrm{I} - \lambda W$ and, $B = \mathrm{I} - \rho W$ which return respectively the spatial filter of the response and the error.

UNCORRECTED PROOFS

UNCORRECTED PROOFS