Practicals 1

Giovanni Millo

Abstract

Keywords: panel data, interfaces, formula, model, pooled OLS, robust covariances, R.

This course notes follow the style and typographical conventions of the *Journal of Statistical Software*, particularly as regards `code` and **package names**. R code and output are printed as follows:

```
> print("hello")

[1] "hello"
```

1. The plm package

Load the **plm** package (estimators and tests for panel data, plus some data infrastructure) and dependencies

```
> library(plm)
>
```

1.1. Basic data conventions

The data conventions in **plm** are not very strict. You have as much freedom as you have responsibility. For example, you can use standard `data.frame`s, but then *it is assumed that the individual and time indices are in the two first columns, in this order*. Alternatives are:

- specify indices at runtime
- transform into a `pdata.frame`

R often works like this: you can take shortcuts if you know what you're doing, but then you must not complain if anything goes wrong. You can specify arguments only by order, assuming a function knows that, e.g., `formula` goes first; and you can nest many statements instead of executing step by step. Nevertheless, it is generally advisable to stay on the safe side.

Load the example dataset `Grunfeld`[1] and inspect it, referring also to the online documentation:

```
> data(Grunfeld)
> head(Grunfeld)
```

```
  firm year   inv  value capital
1    1 1935 317.6 3078.5     2.8
2    1 1936 391.8 4661.7    52.6
3    1 1937 410.6 5387.1   156.9
4    1 1938 257.7 2792.2   209.2
5    1 1939 330.8 4313.2   203.4
6    1 1940 461.2 4643.9   207.2
```

```
> ?Grunfeld
```

This dataset is an object of class `data.frame`, the basic data type for regression analysis in R. Check:

```
> class(Grunfeld)
```

```
[1] "data.frame"
```

```
> str(Grunfeld)
```

```
'data.frame':        200 obs. of  5 variables:
 $ firm   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ year   : int  1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 ...
 $ inv    : num  318 392 411 258 331 ...
 $ value  : num  3078 4662 5387 2792 4313 ...
 $ capital: num  2.8 52.6 156.9 209.2 203.4 ...
```

As the columns are in the right ordering already, the following estimation commands are equivalent:

```
> femod <- plm(formula=inv~value+capital, data=Grunfeld)
> femod1 <- plm(formula=inv~value+capital, data=Grunfeld, index=c("firm","year"))
```

but if we reorder columns in any other order, e.g. totally random,

```
> grun0 <- Grunfeld[, order(rnorm(dim(Grunfeld)[[2]]))]
> head(grun0)
```

[1] Many panel data classes or textbooks start with the Grunfeld example. For an interesting historical review, and reasons why every book reports different results, see Kleiber and Zeileis, *The Grunfeld data at 50*, German Economic Review, 11 (4), 404-417, 2010.

```
   value   inv capital firm year
1 3078.5 317.6     2.8    1 1935
2 4661.7 391.8    52.6    1 1936
3 5387.1 410.6   156.9    1 1937
4 2792.2 257.7   209.2    1 1938
5 4313.2 330.8   203.4    1 1939
6 4643.9 461.2   207.2    1 1940
```

then the only way is to tell R which are the indices.

R-tip: **specifying indices will save you a lot of trouble**. In general, calling every argument by name is the safest strategy.

1.2. The `pdata.frame`

A more advanced possibility is to define data as an object of more specific type: the `pdata.frame`, inheriting from `data.frame`s but adding more features:

```
> pgrun <- pdata.frame(Grunfeld, index=c("firm", "year"))
> class(pgrun)

[1] "pdata.frame" "data.frame"

> str(pgrun)

Classes âĂŸpdata.frameâĂŹ and 'data.frame':        200 obs. of  5 variables:
 $ firm   : Factor w/ 10 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "names")= chr [1:200] "1-1935" "1-1936" "1-1937" "1-1938" ...
  ..- attr(*, "index")=Classes âĂŸpindexâĂŹ and 'data.frame':        200 obs. of  2 variab
  .. ..$ firm: Factor w/ 10 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
  .. ..$ year: Factor w/ 20 levels "1935","1936",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ year   : Factor w/ 20 levels "1935","1936",..: 1 2 3 4 5 6 7 8 9 10 ...
  ..- attr(*, "names")= chr [1:200] "1-1935" "1-1936" "1-1937" "1-1938" ...
  ..- attr(*, "index")=Classes âĂŸpindexâĂŹ and 'data.frame':        200 obs. of  2 variab
  .. ..$ firm: Factor w/ 10 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
  .. ..$ year: Factor w/ 20 levels "1935","1936",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ inv    : 'pseries' Named num  318 392 411 258 331 ...
  ..- attr(*, "names")= chr [1:200] "1-1935" "1-1936" "1-1937" "1-1938" ...
  ..- attr(*, "index")=Classes âĂŸpindexâĂŹ and 'data.frame':        200 obs. of  2 variab
  .. ..$ firm: Factor w/ 10 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
  .. ..$ year: Factor w/ 20 levels "1935","1936",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ value  : 'pseries' Named num  3078 4662 5387 2792 4313 ...
  ..- attr(*, "names")= chr [1:200] "1-1935" "1-1936" "1-1937" "1-1938" ...
  ..- attr(*, "index")=Classes âĂŸpindexâĂŹ and 'data.frame':        200 obs. of  2 variab
  .. ..$ firm: Factor w/ 10 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
  .. ..$ year: Factor w/ 20 levels "1935","1936",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ capital: 'pseries' Named num  2.8 52.6 156.9 209.2 203.4 ...
```

```
  ..- attr(*, "names")= chr [1:200] "1-1935" "1-1936" "1-1937" "1-1938" ...
  ..- attr(*, "index")=Classes âĂŸpindexâĂŹ and 'data.frame':        200 obs. of  2 variab
  .. ..$ firm: Factor w/ 10 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
  .. ..$ year: Factor w/ 20 levels "1935","1936",..: 1 2 3 4 5 6 7 8 9 10 ...
 - attr(*, "index")=Classes âĂŸpindexâĂŹ and 'data.frame':        200 obs. of  2 variables
  ..$ firm: Factor w/ 10 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
  ..$ year: Factor w/ 20 levels "1935","1936",..: 1 2 3 4 5 6 7 8 9 10 ...
```

```
> summary(pgrun)
```

```
      firm         year          inv               value
 1      :20   1935   : 10   Min.   :   0.93   Min.   :  58.12
 2      :20   1936   : 10   1st Qu.:  33.56   1st Qu.: 199.97
 3      :20   1937   : 10   Median :  57.48   Median : 517.95
 4      :20   1938   : 10   Mean   : 145.96   Mean   :1081.68
 5      :20   1939   : 10   3rd Qu.: 138.04   3rd Qu.:1679.85
 6      :20   1940   : 10   Max.   :1486.70   Max.   :6241.70
 (Other):80   (Other):140
    capital
 Min.   :   0.80
 1st Qu.:  79.17
 Median : 205.60
 Mean   : 276.02
 3rd Qu.: 358.10
 Max.   :2226.30
```

```
> class(pgrun$capital)
```

```
[1] "pseries" "numeric"
```

```
> str(pgrun$capital)
```

```
 'pseries' Named num [1:200] 2.8 52.6 156.9 209.2 203.4 ...
 - attr(*, "names")= chr [1:200] "1-1935" "1-1936" "1-1937" "1-1938" ...
 - attr(*, "index")=Classes âĂŸpindexâĂŹ and 'data.frame':        200 obs. of  2 variables
  ..$ firm: Factor w/ 10 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
  ..$ year: Factor w/ 20 levels "1935","1936",..: 1 2 3 4 5 6 7 8 9 10 ...
```

Notice the class of individual series extracted. `data.frame`s are ok for ordinary estimation. Having put the data into a `pdata.frame` also allows for lagging and differencing:

```
> capital.1 <- lag(pgrun$capital)
> d.capital <- diff(pgrun$capital)
> head(capital.1)
```

```
1-1935 1-1936 1-1937 1-1938 1-1939 1-1940
    NA    2.8   52.6  156.9  209.2  203.4
```

```
> head(d.capital)

1-1935 1-1936 1-1937 1-1938 1-1939 1-1940
    NA   49.8  104.3   52.3   -5.8    3.8

> head(pgrun$capital)

1-1935 1-1936 1-1937 1-1938 1-1939 1-1940
   2.8   52.6  156.9  209.2  203.4  207.2
```

Although they can be useful in manipulating data, `lag` and `diff` operators are more often used in formulas, inside estimation statements. In fact, R rarely requires you to do any data transformation in advance, because most can be done inside formulas, keeping the raw data as they are.

1.3. Estimation interface

Standard linear model

Let us quickly review estimation and post-estimation commands. First, define your model:

```
> fm <- inv~value+capital
```

The standard OLS estimator in R (pooling all data together irrespective of firm and time):

```
> olsmod <- lm(formula=fm, data=Grunfeld)
> class(olsmod)

[1] "lm"
```

R does not flood you with output: it creates an object instead which you can inspect at will. Standard and more compact way to look at the coefficients' table:

```
> summary(olsmod)

Call:
lm(formula = fm, data = Grunfeld)

Residuals:
    Min      1Q  Median      3Q     Max
-291.68  -30.01    5.30   34.83  369.45

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -42.714369   9.511676  -4.491 1.21e-05 ***
value         0.115562   0.005836  19.803  < 2e-16 ***
```

```
capital       0.230678   0.025476   9.055  < 2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 94.41 on 197 degrees of freedom
Multiple R-squared:  0.8124,        Adjusted R-squared:  0.8105
F-statistic: 426.6 on 2 and 197 DF,  p-value: < 2.2e-16
```

```
> library(lmtest)
> coeftest(olsmod)

t test of coefficients:

              Estimate  Std. Error t value  Pr(>|t|)
(Intercept) -42.7143694   9.5116760 -4.4907 1.207e-05 ***
value         0.1155622   0.0058357 19.8026 < 2.2e-16 ***
capital       0.2306785   0.0254758  9.0548 < 2.2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

> coef(olsmod)["capital"]

  capital
0.2306785
```

By the way, `coeftest` can do much more: in particular, it can plug a robust covariance matrix into the significance test:

```
> library(sandwich)
> coeftest(olsmod, vcov=vcovHC)

t test of coefficients:

              Estimate  Std. Error t value  Pr(>|t|)
(Intercept) -42.7143694  14.0134955 -3.0481  0.002619 **
value         0.1155622   0.0071627 16.1340 < 2.2e-16 ***
capital       0.2306785   0.0585099  3.9426  0.000112 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1
```

Panel models

An (apparently) equivalent formulation in **plm**:

```
> poolmod <- plm(fm, data=Grunfeld, model="pooling")
> class(poolmod)
```

```
[1] "plm"        "panelmodel"

> coeftest(poolmod)

t test of coefficients:

              Estimate  Std. Error t value  Pr(>|t|)
(Intercept) -42.7143694   9.5116760 -4.4907 1.207e-05 ***
value         0.1155622   0.0058357 19.8026 < 2.2e-16 ***
capital       0.2306785   0.0254758  9.0548 < 2.2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1
```

The object we created has class `plm` and `panelmodel`. This means it retains "memory" of the individual and time indices, it allows structured extraction of the data and more. E.g., it is possible to apply robust panel covariances of the clustered type:

```
> coeftest(poolmod, vcov=vcovHC)

t test of coefficients:

             Estimate Std. Error t value Pr(>|t|)
(Intercept) -42.714369  19.279431 -2.2155 0.027868 *
value         0.115562   0.015003  7.7027 6.35e-13 ***
capital       0.230678   0.080201  2.8763 0.004467 **
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1
```

Why are the resulting SEs different? Because although computationally equivalent in this case, *lm() and plm() produce different object types*, and therefore different methods can be applied to them.

R-tip: **R is object-oriented: what a function does depends on what it is applied to**

Notice that what R does is in both cases the most sensible default. In fact, clustering the covariance by group is always advisable if working with pooled data (see Moulton (1986)).

It is indeed possible to reproduce the OLS results, but one must override some defaults (what was the most sensible, or only option, in OLS is not a sensible default in pooled OLS any more):

```
> coeftest(olsmod, vcov=vcovHC)

t test of coefficients:

              Estimate  Std. Error t value  Pr(>|t|)
(Intercept) -42.7143694  14.0134955 -3.0481  0.002619 **
value         0.1155622   0.0071627 16.1340 < 2.2e-16 ***
capital       0.2306785   0.0585099  3.9426  0.000112 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1
```

```
> coeftest(poolmod, vcov=function(x) vcovHC(x, method="white1", type="HC3"))

t test of coefficients:

              Estimate  Std. Error t value  Pr(>|t|)
(Intercept) -42.7143694  14.0134955 -3.0481  0.002619 **
value         0.1155622   0.0071627 16.1340 < 2.2e-16 ***
capital       0.2306785   0.0585099  3.9426  0.000112 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1
```

Lags and differences can be used directly in formulas, as observed, and will correctly lag and difference the data according to the panel structure. Let us add a lag of the regressand to Grunfeld's specification, using formula updating facilities:

```
> poolAR1mod <- plm(update(fm, .~.+lag(inv)), data=Grunfeld, model="pooling")
```

Analogously, a first difference model can be estimated by:

```
> dfm <- diff(inv)~diff(value)+diff(capital)
> poolfdmod <- plm(dfm, data=Grunfeld, model="pooling")
```

although there is, as we will discover, an easier way to estimate FD models in **plm**:

```
> fdmod <- plm(fm, data=Grunfeld, model="fd")
```

You can now verify that the results are identical. Notice that we are still using standard `data.frames` as all data manipulation occurs inside the estimating functions. Of course, a `pdata.frame` would have suited us equally well.

2. Exercises

2.1. Munnell's productivity model

Munnell (1990), *Public capital productivity*: Does public capital (roads, water facilities, public buildings and structures) help growth? (Example 3 in Baltagi)

48 US states, annual data 1970-1986. Production function:

$$log(gdp) = \alpha + \beta_1 log(pcap) + \beta_2 log(pc) + \beta_3 log(emp) + \beta_4 unemp$$

You are required to:

1. load the *built-in* dataset `Produc`
2. read the online documentation to check variable names etc.

3. inspect the dataset without jamming the output log
4. determine its nature as an R object
5. write the model specification as a `formula` object
6. estimate a pooled specification by OLS both by `lm` and by `plm`, also adding an AR(2) term, and output the results with `summary` and `coeftest`
7. estimate the model on first differences specifying them in the model formula

and also (less straightforward) to:

1. test the hypothesis of constant returns to scale ($\beta_1+\beta_2+\beta_3=1$) by `linearHypothesis` in package **car**
2. remembering that the individual and time indices are regular variables, add
   (a) a time trend
   (b) time dummies

   to the basic model specification, estimate it and print the results. *Hint: a discrete variable can be made into a categoric variable, also called a* `factor` *in* R

Affiliation:
Giovanni Millo
DEAMS, University of Trieste
Piazzale Europa 4
34127 Trieste (Italy)
E-mail: giovanni_millo@deams.units.it