gdata.frame, an R package for grouped data

Yves Croissant

extending the formula/data interface

- **plm** package for panel data econometrics : entities (individuals, firms, countries, . . .) observed for several time periods,
- **mlogit** package for RUM (random utility models or discrete choice models) : choice situation with several alternatives
- limits of the basic formula/data interface :
  - several formulas (for different sets of covariates),
  - the structure of the data set
- two solutions :
  - adding extra arguments in the fitting function,
  - providing enhanced `formulas` and `data.frames`.

enhanced formula

Formula package (Zeileis and Croissant, 2010)

- several responses : `y1 + y2 ~ x1 + x2`
- several sets of covariates : `y ~ x1 + x2 | z1 + z2`

example for instrumental variable estimation :

```
plm(y ~ x1 + x2, data = somedata, instruments = ~ z1 + z2)
```

replaced by :

```
plm(y ~ x1 + x2 | z1 + z2, data = somedata)
```

enhanced data frame in plm and mlogit

- each package provides a function and a class for enhanced `data.frames` :
  - `pdata.frame` for **plm**,
  - `mlogit.data` for **mlogit**
- code duplication,
- inconsistencies,
- weak integration with **tibble** and **dplyr**

Panel example 1 : The Produc data set

48 states nested in 9 regions, 17 years

```
##       state year              region   gsp unemp
## 1   Alabama 1970 East South Central 28418   4.7
## 2   Alabama 1971 East South Central 29375   5.2
## 3   Alabama 1972 East South Central 31303   4.7
## 4   Alabama 1973 East South Central 33430   3.9
## 5   Alabama 1974 East South Central 33749   5.5
## 6   Alabama 1975 East South Central 33604   7.7
## 7   Alabama 1976 East South Central 35764   6.8
## 8   Alabama 1977 East South Central 37463   7.4
## 9   Alabama 1978 East South Central 39964   6.3
## 10  Alabama 1979 East South Central 40979   7.1
## 11  Alabama 1980 East South Central 40380   8.8
```

Panel example 2 : The Wages data set

595 individuals, 7 years (1976-1982)

```
##     exp wks bluecol ind south smsa married  sex union
## 1     3  32      no   0   yes   no     yes male    no
## 2     4  43      no   0   yes   no     yes male    no
## 3     5  40      no   0   yes   no     yes male    no
## 4     6  39      no   0   yes   no     yes male    no
## 5     7  42      no   1   yes   no     yes male    no
## 6     8  35      no   1   yes   no     yes male    no
## 7     9  32      no   1   yes   no     yes male    no
## 8    30  34     yes   0    no   no     yes male    no
## 9    31  27     yes   0    no   no     yes male    no
## 10   32  33     yes   1    no   no     yes male   yes
## 11   33  30     yes   1    no   no     yes male    no
```

RUM example 1 : The RiskyTransport data set

1793 choice situations nested in 561 individuals, 4 alternatives

```
##          id choice        mode     cost       risk chid
## 1   8020605      0  WaterTaxi 59.31009  2.551270    1
## 2   8020605      1      Ferry 34.72887  4.431152    1
## 3   8020605      0 Hovercraft 57.04705  3.881836    1
## 4   8020605      0 Helicopter 99.86929 18.408203    2
## 5   8020605      0  WaterTaxi 59.31009  2.551270    2
## 6   8020605      1      Ferry 34.72887  4.431152    2
## 7   8020605      0 Hovercraft 81.04450  3.881836    2
## 8   8020605      0 Helicopter 99.86929 18.408203    3
## 9   8020605      0  WaterTaxi 59.31009  2.551270    3
## 10  8020605      1      Ferry 34.72887  4.431152    3
## 11  8020605      0 Hovercraft 81.04450  3.881836    3
```

RUM example 2 : The JapaneseFDI data set

452 firms, 57 alternatives (regions) nested in 9 countries

```
##     firm country region choice    scrate      gdp
## 1      3      BE    BE0      0 0.5982960  28592.7
## 2      3      BE    BE1      0 0.5982960  66857.2
## 3      3      BE    BE2      0 0.5982960  31036.9
## 4      3      BE    BE3      0 0.5982960  18236.9
## 5      3      DE    DE1      0 0.2667937  31773.8
## 6      3      DE    DE2      0 0.2667937  37169.3
## 7      3      DE    DE3      0 0.2667937  90500.6
## 8      3      DE    DE5      0 0.2667937  12061.8
## 9      3      DE    DE6      0 0.2667937 237818.7
## 10     3      DE    DE7      0 0.2667937  92198.5
## 11     3      DE    DE9      0 0.2667937  47771.2
```

the gdata.frame package

A small stand alone package with only one exported function `gdata.frame` and a few methods :

1. suitable for data sets where observations are uniquely defined by a combination of two indexes,
2. possible nesting structure for both of the indexes,
3. indexes stored as an attribute of the data frame,
4. extracted series inherit this index attribute.

Other package

- **spacetime** (Pebesma, 2012) : each observation is a spatial entity observed at a specific date,
- **tsibble** (Wang, Cook and Hyndman, 2019) : each observation defined by a combination of an entity (called `key`) and a date (called `index`).

a simple data set

```
##    country year  gdp unemp    zone period
## 1       FR 2000  3.2   8.6    euro natcur
## 2       FR 2001  1.2   7.8    euro natcur
## 3       FR 2002  0.4   7.9    euro   euro
## 4       FR 2003  0.1   8.5    euro   euro
## 5       DK 2000  3.4   4.3 noteuro natcur
## 6       DK 2001  0.5   4.5 noteuro natcur
## 7       DK 2002  0.1   4.6 noteuro   euro
## 8       DK 2003  0.1   5.4 noteuro   euro
## 9       IT 2000  3.7  10.0    euro natcur
## 10      IT 2001  1.7   9.0    euro natcur
## 11      IT 2002  0.0   8.5    euro   euro
## 12      IT 2003 -0.4   8.4    euro   euro
## 13      UK 2000  3.1   5.4 noteuro natcur
## 14      UK 2001  2.4   5.0 noteuro natcur
```

Basic use of the gdata.frame function

- a gdata.frame is created using the gdata.frame function
- first two mandatory arguments, data.frame and index
- index is a list of two characters of length one or two (in case of nesting).
- the returned data.frame is sorted first by the first index (zone and country) and then by the second index (period and year).

```
library("gdata.frame")
geuro <- gdata.frame(euro,
                     index = list(c("country", "zone"),
                                  c("year", "period")))
```

extracting the index

- the index attribute can be extracted using the index function.

```
index(geuro) %>% head
```

```
##    country zone year period
## 1       FR euro 2000 natcur
## 2       FR euro 2001 natcur
## 3       FR euro 2002   euro
## 4       FR euro 2003   euro
## 9       IT euro 2000 natcur
## 10      IT euro 2001 natcur
```

- the index has a ids attribute :

```
attr(index(geuro), "ids")
```

```
## [1] 1 1 2 2
```

One nesting variables

```
gd <- gdata.frame(euro, index = list(c("country"),
                                     c("year", "period")))
head(index(gd))
```

```
##   country year period
## 5      DK 2000 natcur
## 6      DK 2001 natcur
## 7      DK 2002   euro
## 8      DK 2003   euro
## 1      FR 2000 natcur
## 2      FR 2001 natcur
```

```
attr(index(gd), "ids")
```

```
## [1] 1 2 2
```

No nesting variables

```
gd1 <- gdata.frame(euro, index = list("country", "year"))
gd2 <- gdata.frame(euro, index = c("country", "year"))
identical(gd1, gd2)
```

```
## [1] TRUE
```

One index

```
gd1 <- gdata.frame(euro, index = list(c("country", "zone")))
head(index(gd1), 2)
```

```
##   country zone id2
## 1      FR euro   1
## 2      FR euro   2
```

```
attr(index(gd1), "ids")
```

```
## [1] 1 1 2
```

```
gd2 <- gdata.frame(euro, index = "country")
head(index(gd2), 2)
```

```
##   country id2
## 5      DK   1
## 6      DK   2
```

integer index

For "balanced" data only, ordered by the first, and by the second index, index is the number of alternatives of the first index

```
gd1 <- gdata.frame(euro, index = 4)
head(index(gd1))
```

```
##   id1 id2
## 1   1   1
## 2   1   2
## 3   1   3
## 4   1   4
## 5   2   1
## 6   2   2
```

no index

the first two columns are assumed to contain the two indexes :

```
gdata.frame(euro) %>% index %>% head
```

```
##    country year
## 5       DK 2000
## 6       DK 2001
## 7       DK 2002
## 8       DK 2003
## 1       FR 2000
## 2       FR 2001
```

[with one argument or the first argument empty

```
geuro[, c("gdp", "unemp")] %>% head(2)
```

```
##   gdp unemp
## 1 3.2   8.6
## 2 1.2   7.8
```

```
ge1 <- geuro[, c("gdp"), drop = FALSE]
ge2 <- geuro["gdp"]
ge1 %>% head(2)
```

```
##   gdp
## 1 3.2
## 2 1.2
```

[with two arguments

```
geuro[geuro$year == 2002 & geuro$zone == "euro", ]
```

```
##    country year gdp unemp zone period
## 1       FR 2002 0.4   7.9 euro   euro
## 2       IT 2002 0.0   8.5 euro   euro
```

```
geuro[geuro$year == 2002 & geuro$zone == "euro", ] %>% index
```

```
##     country zone year period
## 3        FR euro 2002   euro
## 11       IT euro 2002   euro
```

Extracting a series from a gdata.frame

```
gdp1 <- geuro[, "gdp"]
gdp2 <- geuro[["gdp"]]
gdp3 <- geuro$gdp
```

Extracted series are of class gseries ; they inherit the index attribute from the gdata.frame they come from

```
class(gdp1)
```

```
## [1] "gseries" "numeric"
```

```
identical(index(gdp1), index(geuro))
```

```
## [1] TRUE
```

User-defined class for extracted series

Extracted series in **plm** of class `pseries` :

```
geuro <- gdata.frame(euro,
                     index = c("country", "year"),
                     clseries = "pseries")
class(geuro$gdp)
```

```
## [1] "pseries" "gseries" "numeric"
```

Special methods available, especially `lag` and `diff` :

```
rbind(geuro$gdp,
      stats::lag(geuro$gdp))[, 1:8]
```

```
##         5   6   7   8   1   2   3   4
## [1,] 3.4 0.5 0.1 0.1 3.2 1.2 0.4 0.1
## [2,]  NA 3.4 0.5 0.1  NA 3.2 1.2 0.4
```

coerce euro to a tibble

```
teuro <- as_tibble(euro)
class(teuro)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
print(teuro, n = 5)
```

```
## # A tibble: 16 x 6
##   country  year   gdp unemp zone    period
##   <chr>    <dbl> <dbl> <dbl> <chr>   <fct>
## 1 FR        2000   3.2   8.6 euro    natcur
## 2 FR        2001   1.2   7.8 euro    natcur
## 3 FR        2002   0.4   7.9 euro    euro
## 4 FR        2003   0.1   8.5 euro    euro
## 5 DK        2000   3.4   4.3 noteuro natcur
## # ... with 11 more rows
```

gdata.frame returns an "indexed" tibble

```
gteuro <- gdata.frame(teuro, index = c("country", "year"))
class(gteuro)
```

```
## [1] "gdata.frame" "tbl_df"      "tbl"         "data.frame"
```

```
print(index(gteuro), n = 4)
```

```
## # A tibble: 16 x 2
##   country year
##   <chr>   <fct>
## 1 DK      2000
## 2 DK      2001
## 3 DK      2002
## 4 DK      2003
## # ... with 12 more rows
```

problems with dplyr

- **dplyr**'s functions returns a `tibble` or a `data.frame`,
- for a `gdata.frame` object :
  - the `class` is lost,
  - the `index` attribute is lost.

the general strategy

```
mutate.gdata.frame <- function(.data, ...){
    attrs <- attributes(.data)
    .data <- as.data.frame(.data)
    .data <- mutate(.data, ...)
    attrs$names <- names(.data)
    attributes(.data) <- attrs
    .data
}
```

- first save the original attributes,
- coerce to a tibble/data.frame,
- use **dplyr**'s function,
- change some of the attributes if necessary,
- attach these attributes to the data.frame and returns the result.