Package ‘lfe’

February 15, 2013

Version 1.4-771

Date 2013-01-16

Title Linear Group Fixed Effects

Author Simen Gaure, Ragnar Frisch Centre for Economic Research

Maintainer Simen Gaure <Simen.Gaure@usit.uio.no>

Copyright 2011, Simen Gaure

Depends Matrix, R (>= 2.10.1), Rcgmin

Suggests compiler

Description Estimate linear models with multiple group fixed effects

Keywords Fixed Effect Estimator, Two-Way Fixed Effects, Panel Data, Multiple Fixed Effects, Kaczmarz Method, Alternating Projections

License Artistic-2.0

Classification/JEL C13, C33, C60

Classification/MSC 62J05, 65F10, 65F50

Repository CRAN

Date/Publication 2013-01-16 12:53:31

NeedsCompilation yes
Description

The package uses the Method of Alternating Projections to estimate linear models with multiple group fixed effects. A generalization of the within estimator. It is thread-parallelized and intended for large problems.

Details

This package is intended for linear models with multiple group fixed effects, i.e. with 2 or more factors with a large number of levels. It performs no other functions than \texttt{lm}, but it uses a special method for projecting out multiple group fixed effects from the normal equations, hence it is faster. It is a generalization of the within estimator. This may be required if the groups have high cardinality (many levels), resulting in tens or hundreds of thousands of dummy-variables. It is also useful if one only wants to control for the group effects, without actually computing them. The package may optionally compute standard errors for the group effects, but this is a very time- and memory-consuming process compared to finding the point estimates.

The estimation is done in two steps. First the other coefficients are estimated with the function \texttt{felm} by centering on all the group means, followed by an OLS (similar to \texttt{lm}). Then the group effects are extracted (if needed) with the function \texttt{getfe}. This method is described in Gaure (2011).

There’s also a function \texttt{demeanlist} which just does the centering on an arbitrary matrix, and there’s a function \texttt{compfactor} which computes the connected components (which are used for interpreting the group effects when there are only two factors, see the Abowd et al references), they are also returned by \texttt{getfe}).

The centering on the means is done with a tolerance which is set by \texttt{options(lfe.eps=1e-8)} (the default). This is a somewhat conservative tolerance, in many cases I’d guess 1e-6 may be sufficient. This may speed up the centering. In the other direction, setting \texttt{options(lfe.eps=0)} will provide maximum accuracy at the cost of computing time and warnings about convergence failure.

The package is threaded, that is, it may use more than one cpu. The number of threads is fetched upon loading the package, from the environment variable \texttt{LFE_THREADS} (or \texttt{OMP_NUM_THREADS}) and stored by \texttt{options(lfe.threads=n)}. This option may be changed prior to calling \texttt{felm}, if so
desired. Note that, typically, \texttt{lfe} is limited by memory-bandwidth, not cpu-speed, thus fast memory and large cache is more important than clock-frequency. It’s therefore also not always true that running on all available cores is much better than running on half of them.

Threading is only done for the centering; the extraction of the group effects is not threaded. The default method for extracting the group coefficients is the iterative Kaczmarz-method, its tolerance is also the \texttt{lfe.eps} option.

For some datasets the Kaczmarz-method is converging very slowly, in this case it may be replaced with the conjugate gradient method of \texttt{Rcgmin} by setting the option options(\texttt{lfe.usecg=TRUE}).

The package has been tested on datasets with approx 20,000,000 observations with 15 covariates and approx 2,300,000 and 270,000 group levels (the \texttt{felm} took about 50 minutes on 8 cpus, the \texttt{getfe} takes 5 minutes). Though, beware that not only the size of the dataset matters, but also its structure.

The package will work with any positive number of grouping factors, but if more than two, their interpretation is in general not well understood, i.e. one should make sure that the coefficients are estimable.

In the exec-directory there is a perl-script \texttt{lfe\_script} which is used at the authors site for creating R-scripts from a simple specification file. The format is documented in \texttt{doc/lfe\_guide.txt}.

\texttt{lfe} is similar in function, though not in method, to the Stata modules \texttt{a2reg} and \texttt{felsdvreg}.

References


Gaure, S. (2011) \textit{OLS with Multiple High Dimensional Category Variables} (to appear)

Ouazad, A. (2008) \textit{A2REG: Stata module to estimate models with two fixed effects}. Statistical Software Components S456942, Boston College Department of Economics. \url{http://ideas.repec.org/c/boc/bocode/s456942.html}

Examples

```r
x <- rnorm(1000)
x2 <- rnorm(length(x))
id <- factor(sample(10,length(x),replace=TRUE))
firm <- factor(sample(3,length(x),replace=TRUE,prob=c(2,1.5,1)))
year <- factor(sample(10,length(x),replace=TRUE,prob=c(2,1.5,rep(1,8))))
id.eff <- rnorm(nlevels(id))
firm.eff <- rnorm(nlevels(firm))
```


year.eff <- rnorm(nlevels(year))
y <- x + 0.25*x2 + id.eff[id] + firm.eff[firm] +
    year.eff[year] + rnorm(length(x))
est <- felm(y ~ x+x2+G(id)+G(firm)+G(year))
summary(est)

ggetfe(est,se=TRUE)

# compare with an ordinary lm
summary(lm(y ~ x+x2+id+firm+year-1))

---

btrap 

**Bootstrap standard errors for the group fixed effects**

**Description**

Bootstrap standard errors for the group fixed effects which were swept out during an estimation
with felm.

**Usage**

btrap(alpha, obj, N=100, ef=NULL,
      eps=getOption('lfe.eps'), threads=getOption('lfe.threads'))

**Arguments**

- **alpha** data frame returned from getfe
- **obj** object of class "felm", usually, a result of a call to felm
- **N** integer. The number of bootstrap iterations
- **ef** function. An estimable function such as in getfe. The default is to use the one
  used on alpha
- **eps** double. Tolerance for centering, as in getfe
- **threads** integer. The number of threads to use

**Details**

The bootstrapping is done in parallel if threads > 1. btrap is run automatically from getfe if
se=TRUE is specified.

**Value**

A data-frame of the same size as alpha is returned, with standard errors filled in.
**Examples**

```r
## create covariates
x <- rnorm(5)
x2 <- rnorm(length(x))

## create individual and firm
id <- factor(sample(1:length(x),replace=TRUE))
f <- factor(sample(5,length(x),replace=TRUE))

## effects
id.eff <- rlnorm(nlevels(id))
f.eff <- rexp(nlevels(f))

## left hand side
y <- x + 0.25*x2 + id.eff[id] + f.eff[f] + rnorm(length(x))

## estimate and print result
est <- felm(y ~ x+x2 + G(id)+G(f))
summary(est)

## extract the group effects
alpha <- getfe(est)
head(alpha)

## bootstrap standard errors
head(btrap(alpha,est))

## bootstrap some differences
ef <- function(v,addnames) {
  w <- c(v[2]-v[1],v[3]-v[2],v[3]-v[1])
  if(addnames) {
    names(w) <-c('id2-id1','id3-id2','id3-id1')
    attr(w,'extra') <- list(note=c('line1','line2','line3'))
  }
  w
}

## check that it's estimable
lfe::is.estimable(ef,est$fe)
head(btrap(alpha,est,ef=ef))
```

---

**compfactor**

*Find the connected components*

**Description**

'compfactor' computes the connected components of the dummy-part of the model.

**Usage**

```r
compfactor(f1)
```
Arguments

\( f_1 \)  

a list of factors defining the dummies

Details

If there are more than two factors, only the first two will be used.

Value

A factor of the same length as the factors in the input argument. It defines the connected components. E.g. \( \text{nlevels}(\text{compfactor}(f_1)) \) will yield the number of connected components.

Examples

```r
## create two factors
f1 <- factor(sample(300,400,replace=TRUE))
f2 <- factor(sample(300,400,replace=TRUE))

## find the components
cf <- compfactor(list(f1=f1,f2=f2))

## show the third largest component
fr <- data.frame(f1,f2,cf)
fr[cf==3,]
```

demeanlist  

Centre vectors on multiple groups

Description

Uses the method of alternating projections to centre a (model) matrix on multiple groups, as specified by a list of factors. This function is called by \texttt{felm}, but it has been made available as standalone in case it’s needed.

Usage

\[
demeanlist(mtx,f1,icpt=icpt,eps=eps,threads=threads,progress=progress)
\]

Arguments

\( mtx \)  

matrix whose columns form vectors to be group-centred. \( mtx \) may also be a list of vectors or matrices.

\( f_1 \)  

list of factors defining the grouping structure

\( \text{icpt} \)  

the position of the intercept, this column is removed from the result matrix

\( \text{eps} \)  

a tolerance for the centering

\( \text{threads} \)  

an integer specifying the number of threads to use

\( \text{progress} \)  

integer. If positive, make progress reports (whenever a vector is centered, but not more often than every \text{progress} minutes)
Details

For each column $y$ in $\text{mtx}$, the equivalent of the following centering is performed, with $\text{cy}$ as the result.

\[
\text{cy} <- y; \text{oldy} <- y-1
\]

while($\sqrt{\text{sum}((\text{cy}-\text{oldy})^2)} >= \text{eps}$) {
    \[
    \text{oldy} <- \text{cy}
    \]
    for($f$ in $\text{fl}$) $\text{cy} <- \text{cy} - \text{tapply}(\text{cy},f,\text{mean})[f]
}

Value

If $\text{mtx}$ is a matrix, a matrix of the same shape, possibly with column $\text{icpt}$ deleted. If $\text{mtx}$ is a list of vectors and matrices, a list of the same length is returned, with the same vector and matrix-pattern, but the matrices have the column $\text{icpt}$ deleted.

Note

In the case that the design-matrix is too large for R, i.e. with more than 2 billion entries, it is possible to create a list of column-vectors instead (provided the vector-length is smaller than 2 billion). $\text{demeanlist}$ will be able to centre these vectors.

Examples

```r
## create a 15x3 matrix
\text{mtx} <- \text{matrix(rnorm(45),15,3)}

## a list of factors
\text{fl} <- \text{list(g1=factor(sample(2,\text{nrow(mtx)},\text{replace=TRUE})),
                     g2=factor(sample(3,\text{nrow(mtx)},\text{replace=TRUE})))}

## centre on both means and print result
\text{mtx0} <- \text{demeanlist(mtx,fl)}
\text{cbind(mtx0,gl=fl[[1]],g2=fl[[2]],comp=compfactor(fl))}

for(i in 1:ncol(mtx0))
    for(n in names(fl))
        cat('col',i,'group',n,'level sums: ',\text{tapply(mtx0[,i],fl[[n]],mean)},'\n')
```

---

**efactory**

Create estimable function

Description

Creates an estimable function for a factor-structure.
Usage

`efactory(obj, opt=NULL, ...)`

Arguments

`obj` 
object of class "felm", usually, a result of a call to `felm`

`opt` 
character. Which type of estimable function

`...` 
various

Details

There are several possibilities for the input parameter `opt`.

- "ref" yields an estimable function which is similar to the default one in `lm`, one reference is forced to 0 in each connected component.
- "zm" Similar to "ref", but the factor which does not contain a reference is made to have zero mean, and an intercept is added.
- "zm2" Similar to "zm", but both factors are made to have zero mean.
- "ln" Least norm function. This will yield the raw coefficients from the Kaczmarz-method, i.e. the solution with smallest norm. This function is not estimable.

Note that in the case with more than two factors, it is not known how to analyze the factors to find the structure of the rank-deficiencies, i.e. the estimable functions. In this case, the factors beyond the first two are assumed not to contribute to the rank-deficiency beyond a single dimension in each. Both "ref" and "zm" keep one such reference at zero in each of these factors. This is the common method when using dummies.

Value

A function of two parameters `function(v, addnames)`. An estimable function (i.e. the result is the vector of some length `N`) of the input vector `v`. When `addnames==TRUE` the returned vector should have names, and optionally an attribute "extra" which is a list of vectors of length `N` which may be used to code additional information.

Note

The author is open to suggestions for other estimable functions, i.e. other useful normalizations of the solutions.

It is not strictly necessary that the `obj` argument is of class "felm", any list with entries "fe" and "cfactor" of the appropriate form will do. That is, `list(fe=f1, cfactor=compfactor(f1))` where `f1` is the list of factors defining the component structure. I.e. if the model is `y ~ ... G(id) + G(firm)`, we have `f1=list(id=id, firm=firm)`. 
Examples

```r
id <- factor(sample(5000,50000,replace=TRUE))
firm <- factor(sample(3000,50000,replace=TRUE))
fl <- list(id=id,firm=firm)
obj <- list(fe=fl,cfactor=compfactor(fl))
## the trivial least-norm transformation, which by the way is non-estimable
print(ef <- efactory(obj,'ln'))
is.estimable(ef,fl)
## then the default
print(ef <- efactory(obj,'ref'))
is.estimable(ef,fl)
# get the names of the coefficients, i.e. the nm-variable in the function
head(evalq(nm,environment(ef)))
```

**felm**  
*Fitting linear models with multiple group fixed effects*

**Description**

'felm' is used to fit linear models with multiple group fixed effects, similarly to `lm`. It uses the Method of Alternating projections to sweep out multiple group effects from the normal equations before estimating the remaining coefficients with OLS.

This function is intended for use with large datasets with multiple group effects of large cardinality. If dummy-encoding the group effects results in a manageable number of coefficients, you are probably better off by using `lm`.

**Usage**

```r
felm(formula,data,iv=NULL,clustervar=NULL)
```

**Arguments**

- `formula`: an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. Similarly to 'lm'. Grouping factors are coded as G(f) (with a capital G).
- `data`: a data frame containing the variables of the model
- `iv`: a formula describing an instrumented variable. Estimated via two step OLS
- `clustervar`: a string or factor. Either the name of a variable or a factor. Used for computing clustered standard errors.

**Value**

`felm` return an object of class "felm". The content of this is currently not documented, but the generic summary-method will yield a summary which may be print’ed.
Note

The standard errors are adjusted for the reduced degrees of freedom coming from the dummies which are implicitly present. In the case of two factors, the exact number of implicit dummies is easy to compute. If there are more factors, the number of dummies is estimated by assuming there’s one reference-level for each factor, this may be a slight over-estimation, leading to slightly too large standard errors.

For the IV-option, it is only necessary to include the instruments on the right hand side. The other covariates, from `formula`, are added automatically in the first step. See the examples.

Ideally, the `clustervar` should have been an option to the `summary`-function instead. However, this would require keeping a copy of the data-matrix in the returned structure. Since this function is intended for very large datasets, we discard the data-matrix to save memory, keeping only residuals and other summary statistics.

Note that the syntax of the `felm`-function has changed, it does no longer allow a separate specification of the group factors, they must be specified with the `G()`-syntax. The old `felm` is still available as `lfe:::felm.old()`, but it will no longer be maintained.

See Also

`getfe`

Examples

```r
## create covariates
x <- rnorm(1000)
x2 <- rnorm(length(x))

## individual and firm
id <- factor(sample(20,length(x),replace=TRUE))
firm <- factor(sample(13,length(x),replace=TRUE))

## effects for them
id.eff <- rnorm(nlevels(id))
firm.eff <- rnorm(nlevels(firm))

## left hand side
u <- rnorm(length(x))
y <- x + 0.5*x2 + id.eff[id] + firm.eff[firm] + u

## estimate and print result
est <- felm(y ~ x+x2+G(id)+G(firm))
summary(est)

## compare with lm
summary(lm(y ~ x + x2 + id + firm-1))

## alternatively
## Not run: felm(y ~ x + x2,fl=list(id=id,firm=firm))
getfe(est)

## End(Not run)

## make an iv-example, Q is instrumented by x3, report robust s.e.
```
getfe

Retrieve the group fixed effects

Description

Compute the group fixed effects, i.e. the dummy parameters, which were swept out during an estimation with \texttt{felm}.

Usage

\begin{verbatim}
getfe(obj, references=NULL, se=FALSE, method='kaczmarz', ef='ref', bN=100)
\end{verbatim}

Arguments

- \texttt{obj} object of class "felm", usually, a result of a call to \texttt{felm}
- \texttt{references} a vector of strings. If there are more than two factors and you have prior knowledge of what the reference levels should be like \texttt{references='id.23'}. Not used with \texttt{method='kaczmarz'
- \texttt{se} logical. Set to \texttt{TRUE} if standard errors for the group effects are wanted. This is very time-consuming for large problems, so leave it as \texttt{FALSE} unless absolutely needed.
- \texttt{method} character string. Either \texttt{'cholesky'}, or the default \texttt{'kaczmarz'}. The latter is often very fast and consumes little memory, it requires an estimable function to be specified, see \texttt{efactory}. The \texttt{'cholesky'} method is no longer maintained as the author sees no use for it.
- \texttt{ef} function. A function of two variables, a vector of group fixed effects and a logical, \texttt{i.e. function(v, addnames)}. This function should be estimable and is used to transform the raw-coefficients \texttt{v} from the kaczmarz-method. The second variable indicates whether the function must return a named vector (if this is \texttt{FALSE}, one may skip the names, saving memory allocations and time).

If a string is specified, it is fed to the \texttt{efactory}-function. The default function is one which picks one reference in each component. Can be set to \texttt{ef=\"ln\"} to yield the minimal-norm solution from the kaczmarz-method.

It can also be set to \texttt{ef=\"zm\"} to get zero means (and intercept) in one of the factors, and a reference in the other.
- \texttt{bN} integer. The number of bootstrap runs when standard errors are requested
Details

For the case with two factors (the f1-argument to felm), one reference in each connected component is adequate when interpreting the results.

For three or more factors, no such easy method is known; for the "cholesky" method-reference levels are found by analyzing the pivoted Cholesky-decomposition of a slightly perturbed system. The "kaczmarz" method provides no rank-deficiency analysis, it is assumed that the factors beyond the two first contribute nothing to the rank-deficiency, so one reference in each is used.

If there are more than two factors, only the first two will be used to report connected components. In this case, it is not known which graph theoretic concept may be used to analyze the rank-deficiency.

The standard errors returned by the Kaczmarz-method are bootstrapped, keeping the other coefficients (from felm) constant, i.e. they are from the variance when resampling the residuals.

Value

The function getfe computes and returns a data frame containing the group fixed effects. It has the columns c('effect','se','obs','comp','fe','idx')

- effect is the estimated effect.
- se is the standard error.
- obs is the number of observations of this level.
- comp is the graph-theoretic component number, useful for interpreting the effects.
- fe is the name of factor.
- idx is the level of the factor.

With the Kaczmarz-method it’s possible to specify a different estimable function.

Examples

```r
## create covariates
x <- rnorm(50000)
x2 <- rnorm(length(x))

## create individual and firm
id <- factor(sample(5000,length(x),replace=TRUE))
firm <- factor(sample(3000,length(x),replace=TRUE))

## effects
id.eff <- rlnorm(nlevels(id))
firm.eff <- rexp(nlevels(firm))

## left hand side
y <- x + 0.25*x2 + id.eff[id] + firm.eff[firm] + rnorm(length(x))

## estimate and print result
est <- felm(y ~ x+x2 + G(id)+G(firm))
summary(est)

## extract the group effects
alpha <- getfe(est,se=TRUE)
```
## find some estimable functions, with standard errors, we don’t get
## names so we must precompute some numerical indices in ef
idx <- match(c('id.5','id.6','firm.11','firm.12'),rownames(alpha))
alpha[idx,]

ef <- function(v,addnames) {
  w <- c(v[idx[[2]]]-v[idx[[1]]],v[idx[[4]]]+v[idx[[1]]],
         v[idx[[4]]]-v[idx[[3]]])
  if(addnames) names(w) <-c('id6-id5','f12+id5','f12-f11')
  w
}
getfe(est,ef=ef,se=TRUE)

## Not run:
summary(lm(y ~ x+x2+id+firm-1))

## End(Not run)

---

## is.estimable

### Verify estimability of function

**Description**

Verify that a function you have written is indeed estimable.

**Usage**

```r
is.estimable(ef,fe,R=NULL)
```

**Arguments**

- `ef`: function. The function to be verified.
- `fe`: list of factors.
- `R`: numeric. Vector of residuals, is NULL, one random is created.

**Value**

Returns a logical.

**Examples**

```r
## create individual and firm
id <- factor(sample(5000,50000,replace=TRUE))
firm <- factor(sample(3000,50000,replace=TRUE))

## create some estimable functions. It's faster to
## use numerical indices in ef rather than strings, and the input v
## to ef has no names, we have to add them when requested
```
ef <- function(v, addnames) {
  w <- c(v[6]-v[5],v[7000]+v[5],v[7000]-v[6000])
  if(addnames) names(w) <- c('id6-id5','f2k+id5','f2k-f1k')
  w
}

is.estimable(ef, list(id=id,firm=firm))

## Then make an error; in the last coordinate, sum two firms
ef <- function(v, addnames) {
  w <- c(v[6]-v[5],v[7000]+v[5],v[7000]+v[6000])
  if(addnames) names(w) <- c('id6-id5','f2k+id5','f2k-f1k')
  w
}

is.estimable(ef, list(id=id,firm=firm))

---

**kaczmarz**  
* Solve a linear system defined by factors *

**Description**

Use the Kaczmarz method to solve a system of the type $Dx = R$, where $D$ is the matrix of dummies created from a list of factors.

**Usage**

```r
kaczmarz(f1, R, eps = getOption('lfe.eps'),
         init = NULL, threads = getOption('lfe.threads'))
```

**Arguments**

- `f1` A list of arbitrary factors of the same length
- `R` numeric. A vector, matrix or list of such of the same length as the factors
- `eps` a tolerance for the method
- `init` numeric. A vector to use as initial value for the Kaczmarz iterations. The algorithm converges to the solution closest to this
- `threads` integer. The number of threads to use when `R` is more than one vector

**Value**

A vector $x$ of length equal to the sum of the number of levels of the factors in `f1`, which solves the system $Dx = R$. If the system is inconsistent, the algorithm may not converge, it will give a warning and return something which may or may not be close to a solution. By setting `eps=0`, maximum accuracy (with convergence warning) will be achieved.
Note

This function is used by `getfe`, it's quite specialized, but it might be useful for other purposes too. In case of convergence problems, setting `options(lfe.usecg=TRUE)` will cause the kaczmarz() function to dispatch to the conjugate gradient method of `Rcgmin`. This may or may not be faster.

Examples

```r
## create factors
f1 <- factor(sample(24000,1000,replace=TRUE))
f2 <- factor(sample(20000,length(f1),replace=TRUE))
f3 <- factor(sample(10000,length(f1),replace=TRUE))
f4 <- factor(sample(8000,length(f1),replace=TRUE))
## the matrix of dummies
D <- t(rBind(as(f1,'sparseMatrix'),as(f2,'sparseMatrix'),
as(f3,'sparseMatrix'), as(f4,'sparseMatrix')))
dim(D)
## an x
truex <- runif(ncol(D))
## and the right hand side
R <- as.vector(D %*% truex)
## solve it
sol <- kaczmarz(list(f1,f2,f3,f4),R)
opt <- options(lfe.usecg=TRUE)
cgsol <- kaczmarz(list(f1,f2,f3,f4),R)
options(opt)
## verify that the solution solves the system Dx = R
res <- D %*% sol - R
cgres <- D %*% cgsol - R
sqrt(sum(res^2))
sqrt(sum(cgres^2))
## but the solution is not equal to the true x, because the system is
## underdetermined
sqrt(sum((sol - truex)^2))
## moreover, the solution from kaczmarz has smaller norm
sqrt(sum(sol^2))
sqrt(sum(truex^2))
```
Index

*Topic models
  felm, 9
  getfe, 11
  lfe-package, 2

*Topic regression
  felm, 9
  getfe, 11
  lfe-package, 2

btrap, 4, 4
compfactor, 2, 5
demeanlist, 2, 6
efactory, 7, 11
felm, 2–4, 6, 8, 9, 11, 12
getfe, 2–4, 10, 11, 15
is.estimable, 13
kaczmarz, 14
lfe (lfe-package), 2
lfe-package, 2
lm, 2, 8, 9