Package 'lfe'

July 2, 2014

Version 1.7-1289

Date 2014-03-13

Title Linear Group Fixed Effects

Author Simen Gaure, Ragnar Frisch Centre for Economic Research

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

Depends R (>= 2.15.2), Matrix (>= 1.1-2)

Imports Formula, xtable

Suggests compiler, igraph, knitr, Rcgmin, stargazer

VignetteBuilder knitr

ByteCompile yes

Description Transforms away factors with many levels prior to doing an OLS.
Useful for estimating linear models with multiple group fixed effects, and for
estimating linear models which uses factors as pure control variables.
Includes support for instrumented variables, robust and multi-way clustered standard errors.

License Artistic-2.0

Classification/JEL C13, C23, C60

Classification/MSC 62J05, 65F10, 65F50

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-03-13 15:30:15

R topics documented:

`lfe-package` *Linear Group Fixed Effects*

Description

The package uses the Method of Alternating Projections to estimate linear models with multiple group fixed effects. A generalization of the within esitmator. 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 `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 estimating them. The package may optionally compute standard errors for the group effects by bootstrapping, but this is a very time- and memory-consuming process compared to finding the point estimates.

As of version 1.6, projecting out interactions between continuous covariates and factors is also supported. I.e. individual slopes, not only individual intercepts.

The estimation is done in two steps. First the other coefficients are estimated with the function `felm` by centering on all the group means, followed by an OLS (similar to lm). Then the group effects are extracted (if needed) with the function `getfe`. This method is described in *Gaure (2013)*, but also appears in *Guimaraes and Portugal(2010)*, disguised as the Gauss-Seidel algorithm.

There's also a function `demeanlist` which just does the centering on an arbitrary matrix, and there's a function `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 `getfe`.

The centering on the means is done with a tolerance which is set by `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 `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 `LFE_THREADS`, `OMP_THREAD_LIMIT`, `OMP_NUM_THREADS` or `NUMBER_OF_PROCESSORS` (for Windows), and stored by `options(lfe.threads=n)`. This option may be changed prior to calling `felm`, if so desired. Note that, typically, **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 `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 **Rcgmin** by setting the option `options(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 `felm` took about 50 minutes on 8 cpus, the `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 `lfescript` which is used at the author's site for creating R-scripts from a simple specification file. The format is documented in `doc/lfeguide.txt`.

lfe is similar in function, though not in method, to the Stata modules `a2reg` and `felsdvreg`.

References

Abowd, J.M., F. Kramarz and D.N. Margolis (1999) *High Wage Workers and High Wage Firms*, Econometrica 67 (1999), no. 2, 251–333. http://dx.doi.org/10.1111/1468-0262.00020

Abowd, J.M., R. Creecy and F. Kramarz (2002) *Computing Person and Firm Effects Using Linked Longitudinal Employer-Employee Data.* Technical Report TP-2002-06, U.S. Census Bureau. http://lehd.did.census.gov/led/library/techpapers/tp-2002-06.pdf

Andrews, M., L. Gill, T. Schank and R. Upward (2008) *High wage workers and low wage firms: negative assortative matching or limited mobility bias?* J.R. Stat. Soc.(A) 171(3), 673–697. http://dx.doi.org/10.1111/j.1467-985X.2007.00533.x

Cornelissen, T. (2008) *The stata command felsdvreg to fit a linear model with two high-dimensional fixed effects.* Stata Journal, 8(2):170–189, 2008. http://econpapers.repec.org/RePEc:tsj:stataj:v:8:y:2008:i:2:p:170-189

Gaure, S. (2013) *OLS with Multiple High Dimensional Category Variables.* Computational Statistics and Data Analysis, 66:8–18, 2013 http://dx.doi.org/10.1016/j.csda.2013.03.024

Gaure, S. (2014) `lfe: Linear Group Fixed Effects`. The R Journal, 5(2):104-117, Dec 2013. http://journal.r-project.org/archive/2013-2/gaure.pdf

Guimaraes, P. and Portugal, P. (2010) *A simple feasible procedure to fit models with high-dimensional fixed effects.* The Stata Journal, 10(4):629–649, 2010. http://www.stata-journal.com/article.html?article=st0212

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

Examples

```
oldopts <- options(lfe.threads=1)
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 | id + firm + year)
summary(est)

getfe(est,se=TRUE)
# compare with an ordinary lm
summary(lm(y ~ x+x2+id+firm+year-1))
options(oldopts)
```

---

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'),
      robust=FALSE, cluster=NULL)
```

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
robust	logical. Should heteroskedastic standard errors be estimated?
cluster	logical or factor. Estimate clustered standard errors.

Details

The bootstrapping is done in parallel if `threads > 1`. `btrap` is run automatically from `getfe` if `se=TRUE` is specified. To save some overhead, the individual iterations are grouped together, the memory available for this grouping is fetched with `getOption('lfe.bootmem')`, which is initialized upon loading of **lfe** to `options(lfe.bootmem=500)` (MB).

If `robust=TRUE`, heteroskedastic robust standard errors are estimated. If `robust=FALSE` and `cluster=TRUE`, clustered standard errors with the cluster specified to `felm()` are estimated. If `cluster` is a factor, it is used for the cluster definition. `cluster` may also be a list of factors.

Value

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

Examples

```
oldopts <- options(lfe.threads=2)
## create covariates
x <- rnorm(3000)
x2 <- rnorm(length(x))

## create individual and firm
id <- factor(sample(700,length(x),replace=TRUE))
firm <- factor(sample(300,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 | id + firm)
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))
options(oldopts)
```

`compfactor` *Find the connected components*

Description

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

Usage

```
compfactor(fl, WW=FALSE)
```

Arguments

`fl`	a list of factors defining the dummies
`WW`	logical. Use Weeks and Williams components

Details

If there are more than two factors and `WW=FALSE`, only the first two will be used.

If `WW=TRUE` and `length(fl) > 2`, the component structure will be as in "A Note on the Determination of Connectedness in an N-Way Cross Classification" by D.L. Weeks and D.R. Williams, Technometrics, vol 6 no 3, August 1964. I.e. in each component, the coefficients within each factor are comparable, that is, their difference is estimable even if there are more than two factors. That is, one may use one reference in each factor in each component, and interpret the coefficients within a component as usual. This is not an exhaustion of all the estimable functions. There is somewhat more about this in one of the vignettes.

Value

A factor of the same length as the factors in the input argument. It defines the connected components. E.g. `nlevels(compfactor(fl))` will yield the number of connected components.

Examples

```
## 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 felm, but it has been made available as standalone in case it's needed.

Usage

```
demeanlist(mtx, fl, icpt=0, eps=getOption('lfe.eps'),
           threads=getOption('lfe.threads'),
           progress=getOption('lfe.pint'),
           accel=getOption('lfe.accel'),
           randfact=TRUE,
           means=FALSE)
```

Arguments

mtx	matrix whose columns form vectors to be group-centred. mtx may also be a list of vectors or matrices.
fl	list of factors defining the grouping structure
icpt	the position of the intercept, this column is removed from the result matrix
eps	a tolerance for the centering
threads	an integer specifying the number of threads to use
progress	integer. If positive, make progress reports (whenever a vector is centered, but not more often than every progress minutes)
accel	integer. Set to 1 if Gearhart-Koshy acceleration should be done.
randfact	logical. Should the order of the factors be randomized? This may improve convergence.
means	logical. Should the means instead of the demeaned matrix be returned? Setting means=TRUE will return mtx - demeanlist(mtx,...), but without the extra copy.

Details

For each column `y` in `mtx`, the equivalent of the following centering is performed, with `cy` as the result.

```
cy <- y; oldy <- y-1
while(sqrt(sum((cy-oldy)**2)) >= eps) {
  oldy <- cy
  for(f in fl) cy <- cy - ave(cy,f)
}
```

Beginning with version 1.6, each factor in `fl` may contain an attribute `'x'` which is a numeric vector of the same length as the factor. The centering is then not done on the means of each group, but on the projection onto the covariate in each group. That is, with a covariate `x` and a factor `f`, it is like projecting out the interaction `x:f`.

Value

If `mtx` is a matrix, a matrix of the same shape, possibly with column `icpt` deleted. If `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 `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). `demeanlist` will be able to centre these vectors.

The `accel` argument enables Gearhart-Koshy acceleration as described in theorem 3.16 by Bauschke, Deutsch, Hundal and Park in "Accelerating the convergence of the method of alternating projections", Trans. Amer. Math. Soc. 355 pp 3433-3461 (2003).

Examples

```
oldopts <- options(lfe.threads=1)
## create a 15x3 matrix
mtx <- matrix(rnorm(45),15,3)

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

## centre on both means and print result
mtx0 <- demeanlist(mtx,fl)
cbind(mtx0,g1=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:',tapply(mtx0[,i],fl[[n]],mean),'\n')

options(oldopts)
```

`efactory` *Create estimable function*

Description

Creates an estimable function for a factor-structure.

Usage

```
efactory(obj, opt='ref',...)
```

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.

In the case that interactions are specified in the model, i.e. with `x:f` in the second part of the formula, these terms are not analyzed to create an estimable function. Only the pure `f` terms are used for this purpose. It is assumed that the `x:f` terms are all identified. Note that in this case, all the levels of `f` are included.

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=fl,cfactor=compfactor(fl))` where `fl` is the list of factors defining the component structure. I.e. if the model is `y ~ ... |id + firm`, we have `fl=list(id=id,firm=firm)`.

Examples

```
oldopts <- options(lfe.threads=1)
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  transformtion, 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)))
options(oldopts)
```

`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

```
 felm(formula, data, iv=NULL, clustervar=NULL, exactDOF=FALSE,
subset, na.action, contrasts=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'. See Details.
`data`	a data frame containing the variables of the model.

iv	a formula describing an instrumented variable. Estimated via two step OLS. Deprecated, replaced by multi part formula specification.
clustervar	a string or factor. Either the name of a variable or a factor, or a list thereof. Used for computing clustered standard errors. Deprecated, replaced by multi part formula specification.
exactDOF	logical. If more than two factors, the degrees of freedom used to scale the covariance matrix (and the standard errors) is normally estimated. Setting `exactDOF=TRUE` causes `felm` to attempt to compute it, but this may fail if there are too many levels in the factors. `exactDOF='rM'` will use the exact method in `Matrix::rankMatrix()`, but this is slower. If the degrees of freedom for some reason are known, they can be specified like `exactDOF=342772`.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The 'factory-fresh' default is `na.omit`. Another possible value is `NULL`, no action. `na.exclude` is currently not supported.
contrasts	an optional list. See the `contrasts.arg` of `model.matrix.default`.
...	other arguments. Currently not used. The `clustervar` and `iv` arguments will be removed from the argument list at a later time, but will continue to be supported in this field.

Details

The formula specification is a response variable followed by a four part formula. The first part consists of ordinary covariates, the second part consists of factors to be projected out. The third part is an IV-specification. The fourth part is a cluster specification for the standard errors. I.e. something like `y ~ x1 + x2 | f1 + f2 |  (Q|W ~ x3+x4) | clu1 + clu2` where `y` is the response, `x1,x2` are ordinary covariates, `f1,f2` are factors to be projected out, `Q` and `W` are covariates which are instrumented by `x3` and `x4`, and `clu1,clu2` are factors to be used for computing cluster robust standard errors. Parts that are not used should be specified as `0`, except if it's at the end of the formula, where they can be omitted. The parentheses are needed in the third part since `|` has higher precedence than `~`.

Interactions between a covariate `x` and a factor `f` can be projected out with the syntax `x:f`. The terms in the second and fourth parts are not treated as ordinary formulas, in particular it is not possible with things like `y ~ x1 | x*f`, rather one would specify `y ~ x1 + x | x:f + f`.

In older versions of **lfe** the syntax was `felm(y ~ x1 + x2 + G(f1)  + G(f2), iv=list(Q ~ x3+x4, W ~ x3+x4), clust`
This syntax still works.

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. Setting `exactDOF='rM'` computes the exact degrees of freedom with `rankMatrix()` in package **Matrix**. Note that version 1.1-0 of **Matrix** has a bug in `rankMatrix()` for sparse matrices which may cause it to return the wrong value. A fix is underway.

For the iv-part of the formula, it is only necessary to include the instruments on the right hand side. The other explanatory covariates, from the first and second part of `formula`, are added automatically in the first stage regressions. See the examples.

The `contrasts` argument is similar to the one in `lm()`, it is used for factors in the first part of the formula. The factors in the second part are analyzed as part of a possible subsequent `getfe()` call.

The old syntax with a single part formula with the `G()` syntax for the factors to transform away is still supported, as well as the `clustervar` and `iv` arguments, but users are encouraged to move to the new multi part formulas as described here. In an upcoming version of **lfe**, the `clustervar` and `iv` arguments will be moved to the `...` argument list. In the event that you use these arguments, and rewriting to the new syntax is impractical, you should make sure to name them (i.e. not use them as positional arguments). `felm` will issue a warning if these two arguments are not named.

Note that the way missing values (NAs) in IV estimations are handled in **lfe** currently may lead to problems. Missing values are removed independently in the first and second stages. Thus, if the instruments have missing values where the other covariates have not, more observations are removed in the first stage than in the second, leading to problems, confusion and general havoc.

An alternative to clustered standard errors is to project out the cluster factors (put them in the second part of the formula) and use heteroskedastic standard errors.

Note that the F-test which is computed by `summary.felm` is unreliable for robust standard errors.

Value

`felm` returns an object of `class "felm"`. It is quite similar to an `"lm"` object, but not entirely compatible.

The generic `summary`-method will yield a summary which may be `print`'ed. The object has some resemblance to the an `lm` object, and some postprocessing methods designed for `lm` may happen to work. It may however be necessary to coerce the object to succeed with this.

The `"felm"` object is a list containing the following fields:

`coefficients`	a numerical vector. The estimated coefficients.
`N`	an integer. The number of observations
`p`	an integer. The total number of coefficients, including those projected out.
`response`	a numerical vector. The response vector.
`fitted.values`	a numerical vector. The fitted values.
`residuals`	a numerical vector. The residuals of the full system, with dummies.
`r.residuals`	a numerical vector. Reduced residuals, i.e. the residuals resulting from predicting *without* the dummies.
`cfactor`	factor of length N. The factor describing the connected components of the two first terms in the second part of the model formula.
`vcv`	a matrix. The variance-covariance matrix.
`fe`	list of factors. A list of the terms in the second part of the model formula.
`step1`	list of '`felm`' objects for the IV 1. step(s), if used.
`iv1fstat`	numerical vector. For IV 1. steps, F-value for excluded instruments, the number of parameters in restricted model and in the unrestricted model.

References

Cameron, A.C., J.B. Gelbach and D.L. Miller (2011) *Robust inference with multiway clustering*, Journal of Business & Economic Statistics 29 (2011), no. 2, 238–249. http://dx.doi.org/10.1198/jbes.2010.07136

See Also

getfe summary.felm

Examples

```
oldopts <- options(lfe.threads=1)
## 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| id + firm)
summary(est)
## compare with lm
summary(lm(y ~ x + x2 + id + firm-1))


# make a weird example with 'reverse causation'
# Q and W are instrumented by x3 and the factor x4. Report robust s.e.
x3 <- rnorm(length(x))
x4 <- sample(12,length(x),replace=TRUE)

Q <- 0.3*x3 + x + 0.2*x2 + id.eff[id] + 0.3*log(x4) - 0.3*y + rnorm(length(x),sd=0.3)
W <- 0.7*x3 - 2*x + 0.1*x2 - 0.7*id.eff[id] + 0.8*cos(x4) - 0.2*y+ rnorm(length(x),sd=0.6)

# add them to the outcome
y <- y + Q + W

ivest <- felm(y ~ x + x2 | id+firm | (Q|W ~x3|x4))
summary(ivest,robust=TRUE)
## Not run:
# compare with the not instrumented fit:
summary(felm(y ~ x + x2 +Q + W |id+firm))
```

```
## End(Not run)
options(oldopts)
```

---

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 felm.

Usage

```
getfe(obj, references=NULL, se=FALSE,
     method='kaczmarz', ef='ref', bN=100, robust=FALSE, cluster=obj[['clustervar']])
```

Arguments

obj	object of class "felm", usually, a result of a call to felm
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 references='id.23'. Not used with method='kaczmarz'
se	logical. Set to TRUE if standard errors for the group effects are wanted. This is **very** time-consuming for large problems, so leave it as FALSE unless absolutely needed.
method	character string. Either 'cholesky', or the default 'kaczmarz'. The latter is often very fast and consumes little memory, it requires an estimable function to be specified, see efactory. The 'cholesky' method is no longer maintained as the author sees no use for it.
ef	function. A function of two variables, a vector of group fixed effects and a logical, i.e. function(v,addnames). This function should be estimable and is used to transform the raw-coefficients v from the kaczmarz-method. The second variable indicates whether the function must return a named vector (if this is FALSE, one may skip the names, saving memory allocations and time). If a string is specified, it is fed to the efactory-function. The default function is one which picks one reference in each component. Can be set to ef="ln" to yield the minimal-norm solution from the kaczmarz-method. It can also be set to ef="zm" to get zero means (and intercept) in one of the factors, and a reference in the other.
bN	integer. The number of bootstrap runs when standard errors are requested.
robust	logical. Should heteroskedastic standard errors be estimated?
cluster	logical or factor. Estimate clustered standard errors.

Details

For the case with two factors (the terms in the second part of the formula supplied 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. If `robust=TRUE`, heteroskedastic robust standard errors are estimated. If `robust=FALSE` and `cluster=TRUE`, clustered standard errors with the cluster specified to `felm()` are estimated. If `cluster` is a factor, it is used for the cluster definition.

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

```
oldopts <- options(lfe.threads=2)
## create covariates
x <- rnorm(4000)
x2 <- rnorm(length(x))

## create individual and firm
id <- factor(sample(500,length(x),replace=TRUE))
firm <- factor(sample(300,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 | id + 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)
options(oldopts)
## 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

```
is.estimable(ef, fe, R=NULL, nowarn=FALSE, keepdiff=FALSE, threshold=1e-5)
```

Arguments

`ef`	function. The function to be verified.
`fe`	list of factors.
`R`	numeric. Vector of residuals, if `NULL`, a random one is created.
`nowarn`	logical. Set to `TRUE` if `is.estimable` should not throw a warning for non-estimable functions.
`keepdiff`	logical. Return differences between two different runs of the Kaczmarz method.
`threshold`	numeric. Threshold for determining estimability.

Details

`is.estimable()` solves the sparse residual system with the Kaczmarz method, using two different initial values. Then `ef()` is applied to the two solutions. If the value of `ef()` differs by more than `1e-5` in any coordinate, `FALSE` is returned, otherwise `TRUE` is returned. If `keepdiff=TRUE`, the vector of differences is attached as an attribute `'diff'` to the returned logical value. If you have problems with estimability, it is a fair guess that those entries with a difference in absolute values smaller than, say, `1e-5` are estimable, whereas the others are not.

Value

Returns a logical.

Examples

```
oldopts <- options(lfe.threads=1)
## 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), keepdiff=TRUE)
options(oldopts)
```

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

Description

Uses 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

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

Arguments

fl	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 fl, 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

```
  oldopts <- options(lfe.threads=1)
## create factors
  f1 <- factor(sample(24000,100000,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)
## verify that the solution solves the system Dx = R
  res <- D %*% sol - R
  sqrt(sum(res^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))
  options(oldopts)
```

summary.felm *Summarizing felm model fits*

Description

summary method for class "felm".

Usage

```
 ## S3 method for class 'felm'
summary(object, ..., robust=!is.null(object$cse))
```

Arguments

object	an object of class "felm", a result of a call to felm.
robust	logical.
...	further arguments passed to or from other methods.

Value

The function summary.felm returns an object of class "summary.felm". It is quite similar to en "summary.lm" object, but not entirely compatible.

The "summary.felm" object is a list containing the following fields:

residuals	a numerical vector. The residuals of the full system, with dummies.
p	an integer. The total number of coefficients, including those projected out.
coefficients	a p x 4 matrix with columns for the estimated coefficients, their standard errors, t-statistic and corresponding (two-sided) p-value.
rse	residual standard error.
r2	R^2, the fraction of variance explained by the model.
r2adj	Adjusted R^2.
fstat	F-statistic.
pval	P-values.
fe	list of factors. A list of the terms in the second part of the model.
iv1fstat	F-statistic for excluded instruments in 1. step IV, see felm.
iv1pval	P-value for iv1fstat.

Note

The standard errors are adjusted for the reduced degrees of freedom coming from the dummies which are implicitly present. They are also small-sample corrected.

If the `robust` parameter is `FALSE`, the returned object will contain ordinary standard errors. If the `robust` parameter is `TRUE`, clustered standard errors are reported if a cluster was specified in the call to `felm`; if not, heteroskedastic robust standard error are reported. The F-statistic is not reliable for robust and clustered errors, the reason being that the author has not yet figured out how to compute it without the full covariance matrix for the dummies that have been projected out. In this case, the `print` method will report the F-statistic under the assumption of normally distributed residuals (which can be computed from the residuals only), and an F-statistic (via a Wald-statistic) for robust/clustered errors which does not take into account the covariates which have been projected out.

For a 1st stage IV-regression, an F-statistic against the model with omitted instruments is also computed, but this one also assumes that the coefficients for the absorbed dummies are equal, so that the covariance comes from the ordinary covariates only.

Index