Package 'FENmlm'

February 12, 2018

Type Package

Title Fixed Effects Nonlinear Maximum Likelihood Models

Version 2.1.0

Date 2018-02-12

Author Laurent Berge

Maintainer Laurent Berge <laurent.berge@uni.lu>

Imports stats, numDeriv, MASS, Rcpp, graphics, utils, Matrix, parallel

LinkingTo Rcpp

Depends R(>= 2.10)

Description Efficient estimation of multiple fixed-effects maximum likelihood models with, possibly, non-linear in parameters right hand sides. Standard-errors can easily be clustered. It also includes tools to seamlessly export (to Latex) the results of various estimations.

License GPL (>= 2)

LazyData TRUE

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-02-12 12:21:30 UTC

R topics documented:

`FENmlm-package` *Fixed Effects Nonlinear Maximum Likelihood Models*

Description

Efficient estimation of multiple fixed-effects maximum likelihood models with, possibly, non-linear in parameters right hand sides. Standard-errors can easily be clustered. It also includes tools to seamlessly export (to Latex) the results of various estimations.

Details

Package:	FENmlm
Type:	Package
Version:	2.1.0
Date:	2018-02-12
License:	GPL-2
LazyLoad:	yes

This package intends to efficiently estimate fixed-effect maximum likelihood models. The function `femlm` estimates fixed-effect unconditionnal maximum likelihood models with, possibly, non-linear in parameters right hand sides. The ML families currently available are: poisson, negative binomial, logit and Gaussian.

Several features are also included such as the possibility to easily compute different types of standard-errors (including multi-way clustering). It is possible to compare the results of severeal estimations by using the function `res2table`, and to export them to Latex using `res2tex`.

Author(s)

Laurent Berge

Maintainer: Laurent Berge <laurent.berge at uni.lu>

`femlm` *Fixed effects maximum likelihood models*

Description

This function estimates maximum likelihood models (e.g., Poisson or Logit) and is efficient to handle any number of fixed effects (i.e. cluster variables). It further allows for nonlinear in parameters right hand sides.

Usage

```
femlm(fml, data, family = c("poisson", "negbin", "logit", "gaussian"), NL.fml,
  cluster, useAcc = TRUE, start, lower, upper, env, start.init, offset,
  nl.gradient, linear.start = 0, jacobian.method = c("simple",
  "Richardson"), useHessian = TRUE, opt.control = list(), cores = 1,
  debug = FALSE, theta.init, ...)
```

Arguments

`fml`	A formula. This formula gives the linear formula to be estimated (it is similar to a `lm` formula), for example: `fml = z~x+y`. To include cluster variables, you can 1) either insert them in this formula using a pipe (e.g. `fml = z~x+y|cluster1+cluster2`), or 2) either use the argment `cluster`. You can add a non-linear element in this formula by using the argment `NL.fml`. If you want to estimate only a non-linear formula without even the intercept, you can use `fml = z~0` in combination with `NL.fml`.
`data`	A data.frame containing the necessary variables to run the model. The variables of the non-linear right hand side of the formula are identified with this `data.frame` names. Note that no `NA` is allowed in the variables to be used in the estimation.
`family`	Character scalar. It should provide the family. The possible values are "poisson" (Poisson model with log-link, the default), "negbin" (Negative Binomial model with log-link), "logit" (LOGIT model with log-link), "gaussian" (Gaussian model).
`NL.fml`	A formula. If provided, this formula represents the non-linear part of the right hand side (RHS). Note that contrary to the `fml` argument, the coefficients must explicitely appear in this formula. For instance, it can be `~a*log(b*x + c*x^3)`, where `a`, `b`, and `c` are the coefficients to be estimated. Note that only the RHS of the formula is to be provided, and NOT the left hand side.
`cluster`	Character vector. The name/s of a/some variable/s within the dataset to be used as clusters. These variables should contain the identifier of each observation (e.g., think of it as a panel identifier).
`useAcc`	Default is `TRUE`. Whether an acceleration algorithm (Irons and Tuck iterations) should be used to otbain the cluster coefficients when there are two clusters.
`start`	A list. Starting values for the non-linear parameters. ALL the parameters are to be named and given a staring value. Example: `start=list(a=1,b=5,c=0)`. Though, there is an exception: if all parameters are to be given the same starting value, you can use the argument `start.init`.
`lower`	A list. The lower bound for each of the non-linear parameters that requires one. Example: `lower=list(b=0,c=0)`. Beware, if the estimated parameter is at his lower bound, then asymptotic theory cannot be applied and the standard-error of the parameter cannot be estimated because the gradient will not be null. In other words, when at its upper/lower bound, the parameter is considered as 'fixed'.
`upper`	A list. The upper bound for each of the non-linear parameters that requires one. Example: `upper=list(a=10,c=50)`. Beware, if the estimated parameter is at

	his upper bound, then asymptotic theory cannot be applied and the standard-error of the parameter cannot be estimated because the gradient will not be null. In other words, when at its upper/lower bound, the parameter is considered as 'fixed'.
`env`	An environment. You can provide an environement in which the non-linear part will be evaluated. (May be useful for some particular non-linear functions.)
`start.init`	Numeric scalar. If the argument `start` is not provided, or only partially filled (i.e. there remain non-linear parameters with no starting value), then the starting value of all remaining non-linear parameters is set to `start.init`.
`offset`	A formula. An offset can be added to the estimation. It should be a formula of the form (for example) ~0.5*x**2. This offset is linearily added to the elements of the main formula 'fml'. Note that when using the argument 'NL.fml', you can directly add the offset there.
`nl.gradient`	A formula. The user can prodide a function that computes the gradient of the non-linear part. The formula should be of the form `~f0(a1,x1,a2,a2)`. The important point is that it should be able to be evaluated by: `eval(nl.gradient[[2]], env)` where `env` is the working environment of the algorithm (which contains all variables and parameters). The function should return a list or a data.frame whose names are the non-linear parameters.
`linear.start`	Numeric named vector. The starting values of the linear part. Note that you can
`jacobian.method`	Character scalar. Provides the method used to numerically compute the jacobian of the non-linear part. Can be either `"simple"` or `"Richardson"`. Default is `"simple"`. See the help of `jacobian` for more information.
`useHessian`	Logical. Should the Hessian be computed in the optimization stage? Default is `TRUE`.
`opt.control`	List of elements to be passed to the optimization method `nlminb`.
`cores`	Integer, default is 1. Number of threads to be used (accelerates the algorithm via the use of openMP routines). This is particularly efficient for the negative binomial and logit models, less so for Gaussian and Poisson likelihoods (unless for large datasets).
`debug`	Logical. If `TRUE` then the log-likelihood as well as all parameters are printed at each iteration. Default is `FALSE`.
`theta.init`	Positive numeric scalar. The starting value of the dispersion parameter if `family="negbin"`. By default, the algorithm uses as a starting value the theta obtained from the model with only the intercept.
`...`	Not currently used.

Details

This function estimates maximum likelihood models where the conditional expectations are as follows:

Gaussian likelihood:
$$E(Y|X) = X\beta$$

Poisson and Negative Binomial likelihoods:

$$E(Y|X) = \exp(X\beta)$$

where in the Negative Binomial there is the parameter θ used to model the variance as $\mu + \mu^2/\theta$, with μ the conditional expectation. Logit likelihood:

$$E(Y|X) = \frac{\exp(X\beta)}{1 + \exp(X\beta)}$$

When there are one or more clusters, the conditional expectation can be written as:

$$E(Y|X) = h(X\beta + \sum_k \sum_m \gamma_m^k \times C_{im}^k),$$

where $h(.)$ is the function corresponding to the likelihood function as shown before. C^k is the matrix associated to cluster k such that C_{im}^k is equal to 1 if observation i is of category m in cluster k and 0 otherwise.

When there are non linear in parameters functions, we can schematically split the set of regressors in two:

$$f(X, \beta) = X^1\beta^1 + g(X^2, \beta^2)$$

with first a linear term and then a non linear part expressed by the function g. That is, we add a non-linear term to the linear terms (which are $X * beta$ and the cluster coefficients). It is always better (more efficient) to put into the argument `NL.fml` only the non-linear in parameter terms, and add all linear terms in the `fml` argument.

Value

An `femlm` object.

`coef`	The coefficients.
`coeftable`	The table of the coefficients with their standard errors, z-values and p-values.
`loglik`	The loglikelihood.
`iterations`	Number of iterations of the algorithm.
`n`	The number of observations.
`k`	The number of parameters of the model.
`call`	The call.
`NL.fml`	The nonlinear formula of the call.
`linear.formula`	The linear formula of the call.
`ll_null`	Log-likelihood of the null model (i.e. with the intercept only).
`pseudo_r2`	The adjusted pseudo R2.
`naive.r2`	The R2 as if the expected predictor was the linear predictor in OLS.
`message`	The convergence message from the optimization procedures.
`sq.cor`	Squared correlation between the dependent variable and its expected value as given by the optimization.

`hessian`	The Hessian of the parameters.
`expected.predictor`	The expected predictor is the expected value of the dependent variable.
`cov.unscaled`	The variance-covariance matrix of the parameters.
`bounds`	Whether the coefficients were upper or lower bounded. – This can only be the case when a non-linear formula is included and the arguments 'lower' or 'upper' are provided.
`isBounded`	The logical vector that gives for each coefficient whether it was bounded or not. This can only be the case when a non-linear formula is included and the arguments 'lower' or 'upper' are provided.
`se`	The standard-error of the parameters.
`scores`	The matrix of the scores (first derivative for each observation).
`family`	The ML family that was used for the estimation.
`resids`	The difference between the dependent variable and the expected predictor.
`dummies`	The sum of the cluster coefficients for each observation.
`clusterNames`	The names of each cluster.
`id_dummies`	The list (of length the number of clusters) of the cluser identifiers for each observation.
`clusterSize`	The size of each cluster.
`obsRemoved`	In the case there were clusters and some observations were removed because of only 0/1 outcome wirhin a cluster, it gives the row numbers of the observations that were removed.
`clusterRemoved`	In the case there were clusters and some observations were removed because of only 0/1 outcome wirhin a cluster, it gives the list (for each cluster) of the clustr identifiers that were removed.
`theta`	In the case of a negative binomial estimation: the overdispersion parameter.

Author(s)

Laurent Berge

References

For models with multiple fixed-effects:

Gaure, Simen, 2013, "OLS with multiple high dimensional category variables", Computational Statistics & Data Analysis 66 pp. 8–18

On the unconditionnal Negative Binomial model:

Allison, Paul D and Waterman, Richard P, 2002, "Fixed-Effects Negative Binomial Regression Models", Sociological Methodology 32(1) pp. 247–265

See Also

See also `summary.femlm` to see the results with the appropriate standard-errors, `getFE` to extract the cluster coefficients, and the functions `res2table` and `res2tex` to visualize the results of multiple estimations.

Examples

```
#
# Linear examples
#

# Load trade data
data(trade)

# We estimate the effect of distance on trade => we account for 3 cluster effects
# 1) Poisson estimation
est_pois = femlm(Euros ~ log(dist_km)|Origin+Destination+Product, trade)
# alternative formulation giving the same results:
# est_pois = femlm(Euros ~ log(dist_km), trade, cluster = c("Origin", "Destination", "Product"))

# 2) Log-Log Gaussian estimation
est_gaus = femlm(log(Euros+1) ~ log(dist_km)|Origin+Destination+Product, trade, family="gaussian")

# 3) Negative Binomial estimation
est_nb = femlm(Euros ~ log(dist_km)|Origin+Destination+Product, trade, family="negbin")

# Comparison of the results using the function res2table
res2table(est_pois, est_gaus, est_nb)
# Now using two way clustered standard-errors
res2table(est_pois, est_gaus, est_nb, se = "twoway")

# Comparing different types of standard errors
sum_white = summary(est_pois, se = "white")
sum_oneway = summary(est_pois, se = "cluster")
sum_twoway = summary(est_pois, se = "twoway")
sum_threeway = summary(est_pois, se = "threeway")

res2table(sum_white, sum_oneway, sum_twoway, sum_threeway)


#
# Non-linear examples
#

# Generating data for a simple example
n = 100
x = rnorm(n, 1, 5)**2
y = rnorm(n, -1, 5)**2
z = rpois(n, x*y) + rpois(n, 2)
base = data.frame(x, y, z)

# Comparing the results of a 'linear' function using a 'non-linear' call
est0L = femlm(z~log(x)+log(y), base)
est0NL = femlm(z~1, base, NL.fml = ~a*log(x)+b*log(y), start = list(a=0, b=0))
# we compare the estimates with the function res2table
res2table(est0L, est0NL)
```

```
# Generating a non-linear relation
z2 = rpois(n, x + y) + rpois(n, 1)
base$z2 = z2

# Using a non-linear form
est1NL = femlm(z2~0, base, NL.fml = ~log(a*x + b*y), start = list(a=1, b=2), lower = list(a=0, b=0))
# we can't estimate this relation linearily
# => closest we can do:
est1L = femlm(z2~log(x)+log(y), base)

res2table(est1L, est1NL)

# Using a custom Jacobian for the function log(a*x + b*y)
myGrad = function(a,x,b,y){
# Custom Jacobian
s = a*x+b*y
data.frame(a = x/s, b = y/s)
}

est1NL_grad = femlm(z2~0, base, NL.fml = ~log(a*x + b*y), start = list(a=1,b=2),
                     nl.gradient = ~myGrad(a,x,b,y))
```

getFE *Extract the Fixed-Effects from a* `femlm` *estimation.*

Description

This function retrives the fixed effects from a femlm estimation. It is useful only when there are more than one cluster.

Usage

```
getFE(x)
```

Arguments

`x`	A femlm object.

Value

A list containig the vectors of the fixed effects.

Author(s)

Laurent Berge

See Also

See also the main estimation function `femlm`. Use `summary.femlm` to see the results with the appropriate standard-errors, `getFE` to extract the cluster coefficients, and the functions `res2table` and `res2tex` to visualize the results of multiple estimations.

Examples

```
# Bilateral network
nb = 20
n = nb**2
k = nb
id1 = factor(rep(1:k, each=n/k))
id2 = factor(rep(1:(n/k), times=k))
d = rep(rnorm(k)**2, each=n/k)
x = rnorm(n, 1, 5)**2
y = rnorm(n, -1, 5)**2
z = rpois(n, x*y+rnorm(n, sd = 3)**2) + rpois(n, 2)
base = data.frame(x, y, z, id1, id2)

# We want to use the ID's of each observation as a variable: we use the option cluster
est_poisson = femlm(z~log(x)+log(y), base, family="poisson", cluster=c("id1", "id2"))

# To get the FE:
myFE = getFE(est_poisson)
```

`print.femlm` *A print facility for* `femlm` *objects. It can compute different types of standard errors.*

Description

This function is very similar to usual `summary` functions as it provides the table of coefficients along with other information on the fit of the estimation.

Usage

```
## S3 method for class 'femlm'
print(x, ...)
```

Arguments

`x`	A femlm object. Obtained using `femlm`.
`...`	Other arguments to be passed to `summary.femlm`.

Author(s)

Laurent Berge

See Also

See also the main estimation functions femlm and femlm. Use summary.femlm to see the results with the appropriate standard-errors, getFE to extract the cluster coefficients, and the functions res2table and res2tex to visualize the results of multiple estimations.

Examples

```
# The data
n = 100
x = rnorm(n, 1, 5)**2
y = rnorm(n, -1, 5)**2
z = rpois(n, x*y) + rpois(n, 10)
base = data.frame(x, y, z)

# Results of the Poisson..
est_poisson = femlm(z~log(x)+log(y), base, family="poisson")
# .. and of the Negative Binomial
est_negbin = femlm(z~log(x)+log(y), base, family="negbin")

# Displaying the results
print(est_poisson)
print(est_negbin)

# Changing the way the standard errors are computed:
summary(est_poisson, se="white") # similar to print(est_poisson, se="white")
summary(est_negbin, se="white")

#
# Now with fixed-effects
#

# Bilateral network
nb = 20
n = nb**2
k = nb
id1 = factor(rep(1:k, each=n/k))
id2 = factor(rep(1:(n/k), times=k))
x = rnorm(n, 1, 5)**2
y = rnorm(n, -1, 5)**2
z = rpois(n, x*y+rnorm(n, sd = 3)**2)
base = data.frame(x, y, z, id1, id2)

# We want to use the ID's of each observation as a variable: we use the option cluster
est_poisson = femlm(z~log(x)+log(y), base, family="poisson", cluster=c("id1","id2"))
# Displaying the results
est_poisson
# now with twoway clustered santard-errors
summary(est_poisson, "twoway")
```

res2table — *Facility to display the results of multiple* `femlm` *estimations.*

Description

This function aggregates the results of multiple estimations and display them in the form of only one table whose rownames are the variables and the columns contain the coefficients and standard-errors.

Usage

```
res2table(..., se = c("standard", "white", "cluster", "twoway", "threeway",
  "fourway"), cluster, digits = 4, pseudo = TRUE, drop, order,
  convergence = FALSE, signifCode = c(`***` = 0.01, `**` = 0.05, `*` = 0.1),
  subtitles, keepFactors = FALSE)
```

Arguments

`...`	Used to capture different `femlm` objects. Note that any other type of element is discarded. Note that you can give a list of `femlm` objects.
`se`	Character scalar. Which kind of standard error should be prompted: “standard” (default), “White”, “cluster”, “twoway”, “threeway” or “fourway”?
`cluster`	A list of vectors. Used only if `se="cluster"`, “se=twoway”, “se=threeway” or “se=fourway”. The vectors should give the cluster of each observation. Note that if the estimation was run using `cluster`, the standard error is automatically clustered along the cluster given in `femlm`.
`digits`	Integer. The number of digits to be displayed.
`pseudo`	Logical, default is `TRUE`. Should the pseudo R2 be displayed?
`drop`	Character vector. This element is used if some variables are not to be displayed. This should be a regular expression (see `regex` help for more info). There can be more than one regular expression. Each variable satisfying the regular expression will be discarded.
`order`	Character vector. This element is used if the user wants the variables to be ordered in a certain way. This should be a regular expression (see `regex` help for more info). There can be more than one regular expression. The variables satisfying the first regular expression will be placed first, then the order follows the sequence of regular expressions.
`convergence`	Logical, default is `TRUE`. Should the convergence state of the algorithm be displayed?
`signifCode`	Named numeric vector, used to provide the significance codes with respect to the p-value of the coefficients. Default is `c("***"=0.01, "**"=0.05, "*"=0.10)`.
`subtitles`	Character vector of the same lenght as the number of models to be displayed. If provided, subtitles are added underneath the dependent variable name.
`keepFactors`	Logical, default is `FALSE`. By default, when factor variables are contained in the estimation, they are printed as if they were a cluster variable. Put to `TRUE` to display all the coefficients of the factor variables.

Value

Returns a data.frame containing the formatted results.

Author(s)

Laurent Berge

See Also

See also the main estimation function `femlm`. Use `summary.femlm` to see the results with the appropriate standard-errors, `getFE` to extract the cluster coefficients, and the functions `res2table` and `res2tex` to visualize the results of multiple estimations.

Examples

```
n = 100
x = rnorm(n, 1, 5)**2
y = rnorm(n, -1, 5)**2
z = rpois(n, x*y) + rpois(n, 2)
base = data.frame(x, y, z)

# Results of the Poisson..
est_poisson = femlm(z~log(x)+log(y), base, family="poisson")
# .. and of the Negative Binomial
est_negbin = femlm(z~log(x)+log(y), base, family="negbin")

# We export the two results in one Latex table:
res2table(est_poisson, est_negbin)

# Changing the names & significance codes
res2table(est_poisson, est_negbin, dict = c("log(x)" = "First variable (ln)"),
        signifCode = c("a" = 0.001, "$$" = 0.1))
```

`res2tex` *Facility to export the results of multiple* `femlm` *estimations in a Latex table.*

Description

This function aggregates the results of multiple estimations and display them in the form of one Latex table whose rownames are the variables and the columns contain the coefficients and standard-errors.

Usage

```
res2tex(..., se = c("standard", "white", "cluster", "twoway", "threeway",
  "fourway"), cluster, digits = 4, pseudo = TRUE, title, sdBelow = TRUE,
  drop, order, dict, file, append = TRUE, convergence = FALSE,
  signifCode = c(`***` = 0.01, `**` = 0.05, `*` = 0.1), label, aic = FALSE,
  sqCor = FALSE, subtitles, showClusterSize = FALSE, bic = TRUE,
  loglik = TRUE, yesNoCluster = c("Yes", "No"), keepFactors = FALSE)
```

Arguments

`...`	Used to capture different `femlm` objects. Note that any other type of element is discarded. Note that you can give a list of `femlm` objects.
`se`	Character scalar. Which kind of standard error should be prompted: “standard” (default), “White”, “cluster”, “twoway”, “threeway” or “fourway”?
`cluster`	A list of vectors. Used only if `se="cluster"`, “se=twoway”, “se=threeway” or “se=fourway”. The vectors should give the cluster of each observation. Note that if the estimation was run using `cluster`, the standard error is automatically clustered along the cluster given in `femlm`.
`digits`	Integer. The number of digits to be displayed.
`pseudo`	Logical, default is `TRUE`. Should the pseudo R2 be displayed?
`title`	Character scalar. The title of the Latex table.
`sdBelow`	Logical, default is `TRUE`. Should the standard-errors be displayed below the coefficients?
`drop`	Character vector. This element is used if some variables are not to be displayed. This should be a regular expression (see `regex` help for more info). There can be more than one regular expression. Each variable satisfying the regular expression will be discarded.
`order`	Character vector. This element is used if the user wants the variables to be ordered in a certain way. This should be a regular expression (see `regex` help for more info). There can be more than one regular expression. The variables satisfying the first regular expression will be placed first, then the order follows the sequence of regular expressions.
`dict`	A named character vector. If provided, it changes the original variable names to the ones contained in the `dict`. Example: I want to change my variable named "a" to "\$log(a)\$" and "b3" to "\$bonus^3\$", then I used `dict=c(a="$log(a)$",b3="$bonus^3$")`.
`file`	A character scalar. If provided, the Latex table will be saved in a file whose path is `file`.
`append`	Logical, default is `TRUE`. Only used if option `file` is used. Should the Latex table be appended to the existing file?
`convergence`	Logical, default is `TRUE`. Should the convergence state of the algorithm be displayed?
`signifCode`	Named numeric vector, used to provide the significance codes with respect to the p-value of the coefficients. Default is `c("***"=0.01, "**"=0.05, "*"=0.10)`.
`label`	Character scalar. The label of the Latex table.

`aic`	Logical, default is `FALSE`. Should the AIC be displayed?
`sqCor`	Logical, default is `FALSE`. Should the squared correlation be displayed?
`subtitles`	Character vector of the same lenght as the number of models to be displayed. If provided, subtitles are added underneath the dependent variable name.
`showClusterSize`	Logical, default is `FALSE`. If `TRUE` and clusters were used in the models, then the number "individuals" of per cluster is also displayed.
`bic`	Logical, default is `TRUE`.Should the BIC be reported?
`loglik`	Logical, default is `TRUE`. Should the log-likelihood be reported?
`yesNoCluster`	A character vector of lenght 2. Default is `c("Yes", "No")`. This is the message displayed when a given cluster is (or is not) included in a regression.
`keepFactors`	Logical, default is `FALSE`. By default, when factor variables are contained in the estimation, they are printed as if they were a cluster variable. Put to `TRUE` to display all the coefficients of the factor variables.

Value

There is nothing returned, the result is only displayed on the console or saved in a file.

Author(s)

Laurent Berge

See Also

See also the main estimation function `femlm`. Use `summary.femlm` to see the results with the appropriate standard-errors, `getFE` to extract the cluster coefficients, and the functions `res2table` and `res2tex` to visualize the results of multiple estimations.

Examples

```
n = 100
x = rnorm(n, 1, 5)**2
y = rnorm(n, -1, 5)**2
z = rpois(n, x*y) + rpois(n, 2)
base = data.frame(x, y, z)

# Results of the Poisson..
est_poisson = femlm(z~log(x)+log(y), base, family="poisson")
# .. and of the Negative Binomial
est_negbin = femlm(z~log(x)+log(y), base, family="negbin")

# We export the two results in one Latex table:
res2tex(est_poisson, est_negbin)

# Changing the names & significance codes
res2tex(est_poisson, est_negbin, dict = c("log(x)" = "First variable (ln)"),
        signifCode = c("a" = 0.001, "$$" = 0.1))
```

`summary.femlm`	*Summary of a `femlm` object. Computes different types of standard errors.*

Description

This function is similar to `print.femlm`. It provides the table of coefficients along with other information on the fit of the estimation. It can compute different types of standard errors. The new variance covariance matrix is an object returned.

Usage

```
## S3 method for class 'femlm'
summary(object, se = c("standard", "white", "cluster",
  "twoway", "threeway", "fourway"), cluster, dof_correction = FALSE,
  forceCovariance = FALSE, keepBounded = FALSE, ...)
```

Arguments

`object`	A femlm object. Obtained using `femlm`.
`se`	Character scalar. Which kind of standard error should be prompted: “standard” (default), “White”, “cluster”, “twoway”, “threeway” or “fourway”?
`cluster`	A list of vectors. Used only if `se="cluster"`, “se=twoway”, “se=threeway” or “se=fourway”. The vectors should give the cluster of each observation. Note that if the estimation was run using `cluster`, the standard error is automatically clustered along the cluster given in `femlm`.
`dof_correction`	Logical, default is `TRUE`. Should there be a degree of freedom correction to the standard errors of the coefficients?
`forceCovariance`	Logical, default is `FALSE`. In the peculiar case where the obtained Hessian is not invertible (usually because of collinearity of some variables), use this option force the covariance matrix, by using a generalized inverse of the Hessian. This can be useful to spot where possible problems come from.
`keepBounded`	Logical, default is `FALSE`. If `TRUE`, then the bounded coefficients (if any) are treated as unrestricted coefficients and their S.E. is computed (otherwise it is not).
`...`	Not currently used.

Value

It returns a `femlm` object with:

`cov.scaled`	The new variance-covariance matrix (computed according to the argument `se`).
`coeftable`	The table of coefficients with the new standard errors.

Author(s)

Laurent Berge

See Also

See also the main estimation function femlm. Use getFE to extract the cluster coefficients, and the functions res2table and res2tex to visualize the results of multiple estimations.

Examples

```
# The data
n = 100
x = rnorm(n,1,5)**2
y = rnorm(n,-1,5)**2
z = rpois(n,x*y) + rpois(n, 2)
base = data.frame(x,y,z)

# Comparing the results of a 'linear' function
est0L = femlm(z~log(x)+log(y), base, family="poisson")

# Displaying the summary
summary(est0L, se="white")
myWhiteVcov = summary(est0L, se="white")$cov.scaled
```

`trade` *Trade data sample*

Description

This data reports trade information between countries of the European Union (EU15).

Usage

`data(trade)`

Format

`trade` is a data frame with 38,325 observations and 6 variables named `Destination`, `Origin`, `Product`, `Year`, `dist_km` and `Euros`.

- Origin2-digits codes of the countries of origin of the trade flow.
- Destination2-digits codes of the countries of destination of the trade flow.
- ProductsNumber representing the product categories (from 1 to 20).
- YearYears from 2007 to 2016
- dist_kmGeographic distance in km between the centers of the countries of origin and destination.

- EurosThe total amount in euros of the trade flow for the specific year/product category/origin-destination country pair.

Source

This data has been extrated from Eurostat on October 2017.

Index