Package 'FENmlm'

May 16, 2019

Type Package

Title Fixed Effects Nonlinear Maximum Likelihood Models

Version 2.4.2

Date 2019-05-16

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

LinkingTo Rcpp

Depends R(>= 2.10)

Description Efficient estimation of maximum likelihood models with multiple fixed-effects. Standard-errors can easily and flexibly be clustered and estimations exported.

License GPL (>= 2)

LazyData TRUE

RoxygenNote 6.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Laurent Berge [aut, cre]

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

Repository CRAN

Date/Publication 2019-05-16 09:20:03 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

This package efficiently estimates maximum likelihood models with multiple fixed-effect (i.e. large factor variables).

The core function is `femlm` which estimates maximum likelihood models with, possibly, non-linear in parameters right hand sides. The ML families 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)

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

References

Berg\'e, Laurent, 2018, "Efficient estimation of maximum likelihood models with multiple fixed-effects: the R package FENmlm." CREA Discussion Papers, 13 (https://wwwen.uni.lu/content/download/110162/1299525/file/2018_13).

`AIC.femlm` *Aikake's an information criterion*

Description

This function computes the AIC (Aikake's, an information criterion) from a `femlm` estimation.

Usage

```
## S3 method for class 'femlm'
AIC(object, ..., k = 2)
```

Arguments

`object`	An object of class `femlm`. Typically the result of a `femlm` estimation.
`...`	Optionally, more fitted objects.
`k`	A numeric, the penalty per parameter to be used; the default k = 2 is the classical AIC (i.e. `AIC=-2*LL+k*nparams`).

Details

The AIC is computed as:

$$AIC = -2 \times LogLikelihood + k \times nbParams$$

with k the penalty parameter.

You can have more information on this crtierion on `AIC`.

Value

It return a numeric vector, with length the same as the number of objects taken as arguments.

Author(s)

Laurent Berge

See Also

`femlm`, `AIC.femlm`, `logLik.femlm`, `nobs.femlm`.

Examples

```
# two fitted models with different expl. variables:
res1 = femlm(Sepal.Length ~ Sepal.Width + Petal.Length +
            Petal.Width | Species, iris)
res2 = femlm(Sepal.Length ~ Petal.Width | Species, iris)

AIC(res1, res2)
BIC(res1, res2)
```

BIC.femlm *Bayesian information criterion*

Description

This function computes the BIC (Bayesian information criterion) from a femlm estimation.

Usage

```
## S3 method for class 'femlm'
BIC(object, ...)
```

Arguments

object	An object of class femlm. Typically the result of a femlm estimation.
...	Optionally, more fitted objects.

Details

The BIC is computed as follows:

$$BIC = -2 \times LogLikelihood + \log(nobs) \times nbParams$$

with k the penalty parameter.

You can have more information on this crtierion on AIC.

Value

It return a numeric vector, with length the same as the number of objects taken as arguments.

Author(s)

Laurent Berge

See Also

femlm, AIC.femlm, logLik.femlm.

Examples

```
# two fitted models with different expl. variables:
res1 = femlm(Sepal.Length ~ Sepal.Width + Petal.Length +
            Petal.Width | Species, iris)
res2 = femlm(Sepal.Length ~ Petal.Width | Species, iris)

AIC(res1, res2)
BIC(res1, res2)
```

`coef.femlm` *Extracts the coefficients from a femlm fit*

Description

This function extracts the coefficients obtained from a model estimated with `femlm`.

Usage

```
## S3 method for class 'femlm'
coef(object, ...)

## S3 method for class 'femlm'
coefficients(object, ...)
```

Arguments

`object`	An object of class `femlm`. Typically the result of a `femlm` estimation.
`...`	Not currently used.

Details

The coefficients are the ones that have been found to maximize the log-likelihood of the specified model. More information can be found on `femlm` help page.

Note that if the model has been estimated with clusters, to obtain the cluster coefficients, you need to use the function `getFE`.

Value

This function returns a named numeric vector.

Author(s)

Laurent Berge

See Also

femlm, summary.femlm, confint.femlm, vcov.femlm, res2table, res2tex, getFE.

Examples

```
# simple estimation on iris data, clustering by "Species"
res = femlm(Sepal.Length ~ Sepal.Width + Petal.Length +
            Petal.Width | Species, iris)

# the coefficients of the variables:
coef(res)

# the cluster coefficients:
getFE(res)
```

---

confint.femlm *Confidence interval for parameters estimated with femlm*

Description

This function computes the confidence interval of parameter estimates obtained from a model estimated with femlm.

Usage

```
## S3 method for class 'femlm'
confint(object, parm, level = 0.95, se = c("standard",
  "white", "cluster", "twoway", "threeway", "fourway"), cluster,
  dof_correction = FALSE, ...)
```

Arguments

object	An object of class femlm. Typically the result of a femlm estimation.
parm	The parameters for which to compute the confidence interval (either an integer vector OR a character vector with the parameter name). If missing, all parameters are used.
level	The confidence level. Default is 0.95.
se	Character scalar. Which kind of standard error should be computed: "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. For one-way clustering, this argument can directly be a vector (instead of a list). If the estimation has been done

with cluster variables, you can give a character vector of the dimensions over which to cluster the SE.

`dof_correction` Logical, default is `FALSE`. Should there be a degree of freedom correction to the standard errors of the coefficients?

`...` Not currently used.

Value

Returns a data.frame with two columns giving respectively the lower and upper bound of the confidence interval. There is as many rows as parameters.

Author(s)

Laurent Berge

Examples

```
# Load trade data
data(trade)

# We estimate the effect of distance on trade (with 3 cluster effects)
est_pois = femlm(Euros ~ log(dist_km) + log(Year) | Origin + Destination +
                 Product, trade)

# confidence interval with "normal" VCOV
confint(est_pois)

# confidence interval with "clustered" VCOV (w.r.t. the Origin factor)
confint(est_pois, se = "cluster")
```

`diagnostic` *Collinearity diagnostics for femlm objects*

Description

In some occasions, the optimization algorithm of `femlm` may fail to converge, or the variance-covariance matrix may not be available. The most common reason of why this happens is collinearity among variables. This function helps to find out which variable is problematic.

Usage

```
diagnostic(x)
```

Arguments

`x` A `femlm` object obtained from function `femlm`.

Details

This function tests: 1) collinearity with the cluster variables, 2) perfect multi-collinearity between the variables, and 3) identification issues when there are non-linear in parameters parts.

Value

It returns a text message with the identified diagnostics.

Examples

```
# Creating an example data base:
cluster_1 = sample(3, 100, TRUE)
cluster_2 = sample(20, 100, TRUE)
x = rnorm(100, cluster_1)**2
y = rnorm(100, cluster_2)**2
z = rnorm(100, 3)**2
dep = rpois(100, x*y*z)
base = data.frame(cluster_1, cluster_2, x, y, z, dep)

# creating collinearity problems:
base$v1 = base$v2 = base$v3 = base$v4 = 0
base$v1[base$cluster_1 == 1] = 1
base$v2[base$cluster_1 == 2] = 1
base$v3[base$cluster_1 == 3] = 1
base$v4[base$cluster_2 == 1] = 1

# Estimations:

# Collinearity with the cluster variables:
res_1 = femlm(dep ~ log(x) + v1 + v2 + v4 | cluster_1 + cluster_2, base)
diagnostic(res_1)
# => collinearity with cluster identified, we drop v1 and v2
res_1bis = femlm(dep ~ log(x) + v4 | cluster_1 + cluster_2, base)
diagnostic(res_1bis)

# Multi-Collinearity:
res_2 =  femlm(dep ~ log(x) + v1 + v2 + v3 + v4, base)
diagnostic(res_2)

# In non-linear part:
res_3 = femlm(dep ~ log(z), base, NL.fml = ~log(a*x + b*y),
              NL.start = list(a=1, b=1), lower = list(a=0, b=0))
diagnostic(res_3)
```

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, na.rm = FALSE, useAcc = TRUE, NL.start, lower,
  upper, env, NL.start.init, offset, nl.gradient, linear.start = 0,
  jacobian.method = c("simple", "Richardson"), useHessian = TRUE,
  opt.control = list(), cores = 1, verbose = 0, theta.init,
  precision.cluster, itermax.cluster = 10000, itermax.deriv = 5000,
  showWarning = TRUE, ...)
```

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`. To include a non-linear in parameters element, you must use the argment `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. Can also be a matrix.
`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).
`na.rm`	Logical, default is `FALSE`. If the variables necessary for the estimation contain NAs and `na.rm = TRUE`, then all observations containing NA are removed prior to estimation and a warning message is raised detailing the number of observations removed.

useAcc — Default is TRUE. Whether an acceleration algorithm (Irons and Tuck iterations) should be used to otbain the cluster coefficients when there are two or more clusters.

NL.start — (For NL models only) A list of starting values for the non-linear parameters. ALL the parameters are to be named and given a staring value. Example: NL.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 NL.start.init.

lower — (For NL models only) 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 — (For NL models only) 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 — (For NL models only) 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.)

NL.start.init — (For NL models only) Numeric scalar. If the argument NL.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 NL.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 — (For NL models only) 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.

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. See the help page of nlminb for more information.

`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 the Gaussian and Poisson likelihoods (unless for very large datasets).

`verbose` Integer, default is 0. It represents the level of information that should be reported during the optimisation process. If `verbose=0`: nothing is reported. If `verbose=1`: the value of the coefficients and the likelihood are reported. If `verbose=2`: 1 + information on the computing tiime of the null model, the cluster coefficients and the hessian are reported.

`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.

`precision.cluster` Precision used to obtain the fixed-effects (ie cluster coefficients). Defaults to `1e-5`. It corresponds to the maximum absolute difference allowed between two iterations. Argument `precision.cluster` cannot be lower than `10000*.Machine$double.eps`.

`itermax.cluster` Maximum number of iterations in the step obtaining the fixed-effects (only in use for 2+ clusters). Default is 10000.

`itermax.deriv` Maximum number of iterations in the step obtaining the derivative of the fixed-effects (only in use for 2+ clusters). Default is 5000.

`showWarning` Logical, default is `TRUE`. Whether warnings should be displayed (concerns warnings relating to: convergence state, collinearity issues and observation removal due to only 0/1 outcomes or presence of NA values).

`...` 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^k_{im} 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.

To estimate only a non-linear formula without even the intercept, you must exclude the intercept from the linear formula by using, e.g., `fml = z~0`.

The over-dispersion parameter of the Negative Binomial family, theta, is capped at 10,000. If theta reaches this high value, it means that there is no overdispersion.

Value

An `femlm` object.

`coefficients`	The named vector of 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.
`nparams`	The number of parameters of the model.
`call`	The call.
`fml`	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.
`message`	The convergence message from the optimization procedures.
`sq.cor`	Squared correlation between the dependent variable and the expected predictor (i.e. fitted.values) obtained by the estimation.
`hessian`	The Hessian of the parameters.
`fitted.values`	The fitted values are the expected value of the dependent variable for the fitted model: that is $E(Y\|X)$.
`cov.unscaled`	The variance-covariance matrix of the parameters.
`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.
`residuals`	The difference between the dependent variable and the expected predictor.
`sumFE`	The sum of the fixed-effects for each observation.

offset	The offset formula.
NL.fml	The nonlinear formula of the call.
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.
clusterNames	The names of each cluster.
id_dummies	The list (of length the number of clusters) of the cluster 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 within 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 within 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

Berge, Laurent, 2018, "Efficient estimation of maximum likelihood models with multiple fixed-effects: the R package FENmlm." CREA Discussion Papers, 13 (https://wwwen.uni.lu/content/download/110162/1299525/file/2018_13).

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 (with same clusters)
est_gaus = update(est_pois, log(Euros+1) ~ ., family="gaussian")

# 3) Negative Binomial estimation
est_nb = update(est_pois, 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)


#
# Example of Equivalences
#
## Not run:
# equivalence with glm poisson
est_glm <- glm(Euros ~ log(dist_km) + factor(Origin) +
            factor(Destination) + factor(Product), trade, family = poisson)

# coefficient estimates + Standard-error
summary(est_glm)$coefficients["log(dist_km)", ]
est_pois$coeftable

# equivalence with lm
est_lm <- lm(log(Euros+1) ~ log(dist_km) + factor(Origin) +
            factor(Destination) + factor(Product), trade)

# coefficient estimates + Standard-error
```

```
summary(est_lm)$coefficients["log(dist_km)", ]
summary(est_gaus, dof_correction = TRUE)$coeftable

## End(Not run)


#
# Non-linear examples
#

# Generating data for a simple example
n = 100
x = rnorm(n, 1, 5)**2
y = rnorm(n, -1, 5)**2
z1 = rpois(n, x*y) + rpois(n, 2)
base = data.frame(x, y, z1)

# Estimating a 'linear' relation:
est1_L = femlm(z1 ~ log(x) + log(y), base)
# Estimating the same 'linear' relation using a 'non-linear' call
est1_NL = femlm(z1 ~ 1, base, NL.fml = ~a*log(x)+b*log(y), NL.start = list(a=0, b=0))
# we compare the estimates with the function res2table (they are identical)
res2table(est1_L, est1_NL)

# Now generating a non-linear relation (E(z2) = x + y + 1):
z2 = rpois(n, x + y) + rpois(n, 1)
base$z2 = z2

# Estimation using this non-linear form
est2_NL = femlm(z2~0, base, NL.fml = ~log(a*x + b*y),
               NL.start = list(a=1, b=2), lower = list(a=0, b=0))
# we can't estimate this relation linearily
# => closest we can do:
est2_L = femlm(z2~log(x)+log(y), base)

# Difference between the two models:
res2table(est2_L, est2_NL)

# Plotting the fits:
plot(x, z2, pch = 18)
points(x, fitted(est2_L), col = 2, pch = 1)
points(x, fitted(est2_NL), col = 4, pch = 2)


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

est2_NL_grad = femlm(z2~0, base, NL.fml = ~log(a*x + b*y),
                     NL.start = list(a=1,b=2), nl.gradient = ~myGrad(a,x,b,y))
```

`fitted.femlm` *Extracts fitted values from a femlm fit*

Description

This function extracts the fitted values from a model estimated with `femlm`. The fitted values that are returned are the *expected predictor*.

Usage

```
## S3 method for class 'femlm'
fitted(object, type = c("response", "link"), ...)

## S3 method for class 'values.femlm'
fitted(object, type = c("response", "link"), ...)
```

Arguments

`object`	An object of class `femlm`. Typically the result of a `femlm` estimation.
`type`	Character either equal to `"response"` (default) or `"link"`. If `type="response"`, then the output is at the level of the response variable, i.e. it is the expected predictor $E(Y\|X)$. If `"link"`, then the output is at the level of the explanatory variables, i.e. the linear predictor $X \cdot \beta$.
`...`	Not currently used.

Details

This function returns the *expected predictor* of a `femlm` fit. The likelihood functions are detailed in `femlm` help page.

Value

It returns a numeric vector of length the number of observations used to estimate the model.

If `type = "response"`, the value returned is the expected predictor, i.e. the expected value of the dependent variable for the fitted model: $E(Y|X)$. If `type = "link"`, the value returned is the linear predictor of the fitted model, that is $X \cdot \beta$ (remind that $E(Y|X) = f(X \cdot \beta)$).

Author(s)

Laurent Berge

See Also

`femlm`, `resid.femlm`, `predict.femlm`, `summary.femlm`, `vcov.femlm`, `getFE`.

Examples

```
# simple estimation on iris data, clustering by "Species"
res_poisson = femlm(Sepal.Length ~ Sepal.Width + Petal.Length +
                    Petal.Width | Species, iris)

# we extract the fitted values
y_fitted_poisson = fitted(res_poisson)

# Same estimation but in OLS (Gaussian family)
res_gaussian = femlm(Sepal.Length ~ Sepal.Width + Petal.Length +
                    Petal.Width | Species, iris, family = "gaussian")

y_fitted_gaussian = fitted(res_gaussian)

# comparison of the fit for the two families
plot(iris$Sepal.Length, y_fitted_poisson)
points(iris$Sepal.Length, y_fitted_gaussian, col = 2, pch = 2)
```

`formula.femlm` *Extract the formula of a femlm fit*

Description

This function extracts the formula from a `femlm` estimation. If the estimation was done with fixed-effects, they are added in the formula after a pipe ("|"). If the estimation was done with a non linear in parameters part, then this will be added in the formula in between `I()`.

Usage

```
## S3 method for class 'femlm'
formula(x, type = c("full", "linear", "NL"), ...)
```

Arguments

`x`	An object of class `femlm`. Typically the result of a `femlm` estimation.
`type`	A character scalar. Default is `type = "full"` which gives back a formula containing the linear part of the model along with the clusters (if any) and the non-linear in parameters part (if any). If `type = "linear"` then only the linear formula is returned. If `type = "NL"` then only the non linear in parameters part is returned.
`...`	Not currently used.

Value

It returns a formula.

Author(s)

Laurent Berge

See Also

femlm, model.matrix.femlm, update.femlm, summary.femlm, vcov.femlm.

Examples

```
# simple estimation on iris data, clustering by "Species"
res = femlm(Sepal.Length ~ Sepal.Width + Petal.Length +
            Petal.Width | Species, iris)

# formula with the cluster variable
formula(res)
# linear part without the cluster variable
formula(res, "linear")
```

---

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. If the cluster coefficients not regular, then several reference points need to be set, leading to the coefficients to be NOT interpretable. If this is the case, then a warning is raised.

Value

A list containig the vectors of the fixed effects.

If there is more than 1 cluster, then the attribute "References" is created. This is a vector of length the number of clusters, each element contains the number of fixed-effects set as references. By construction, the elements of the first clusters are never set as references. In the presence of regular clusters, there should be Q-1 references (with Q the number of clusters).

Author(s)

Laurent Berge

See Also

plot.femlm.allClusters. 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

```
data(trade)

# We estimate the effect of distance on trade => we account for 3 cluster effects
est_pois = femlm(Euros ~ log(dist_km)|Origin+Destination+Product, trade)

# obtaining the cluster coefficients
fe_trade = getFE(est_pois)

# plotting them
plot(fe_trade)

# plotting only the Products fixed-effects & showing more of them
plot(fe_trade$Product, n=8)
```

---

logLik.femlm *Extracts the log-likelihood*

Description

This function extracts the log-likelihood from a femlm estimation.

Usage

```
## S3 method for class 'femlm'
logLik(object, ...)
```

Arguments

object	An object of class femlm. Typically the result of a femlm estimation.
...	Not currently used.

Details

This function extracts the log-likelihood based on the model fit. You can have more information on the likelihoods in the details of the function femlm.

Value

It returns a numeric scalar.

Author(s)

Laurent Berge

See Also

femlm, AIC.femlm, BIC.femlm, nobs.femlm.

Examples

```
# simple estimation on iris data, clustering by "Species"
res = femlm(Sepal.Length ~ Sepal.Width + Petal.Length +
            Petal.Width | Species, iris)

nobs(res)
logLik(res)
```

---

model.matrix.femlm *Design matrix of a femlm model*

Description

This function creates a design matrix of the linear part of a femlm estimation. Note that it is only the linear part and the cluster variables (which can be considered as factors) are excluded from the matrix.

Usage

```
## S3 method for class 'femlm'
model.matrix(object, data, ...)
```

Arguments

object	An object of class femlm. Typically the result of a femlm estimation.
data	If missing (default) then the original data is obtained by evaluating the call. Otherwise, it should be a data.frame.
...	Not currently used.

Value

It returns a design matrix.

Author(s)

Laurent Berge

See Also

femlm, formula.femlm, update.femlm, summary.femlm, vcov.femlm.

Examples

```
# simple estimation on iris data, clustering by "Species"
res = femlm(Sepal.Length ~ Sepal.Width*Petal.Length +
            Petal.Width | Species, iris)

head(model.matrix(res))
```

---

nobs.femlm *Extract the number of observations form a femlm object*

Description

This function simply extracts the number of obsrvations used to estimate a femlm model.

Usage

```
## S3 method for class 'femlm'
nobs(object, ...)
```

Arguments

object	An object of class femlm. Typically the result of a femlm estimation.
...	Not currently used.

Value

It returns an interger.

Author(s)

Laurent Berge

See Also

See also the main estimation functions 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

```
# simple estimation on iris data, clustering by "Species"
res = femlm(Sepal.Length ~ Sepal.Width + Petal.Length +
            Petal.Width | Species, iris)

nobs(res)
logLik(res)
```

`obs2remove` *Finds observations to be removed from ML estimation with factors/clusters*

Description

For Poisson, Negative Binomial or Logit estimations with fixed-effects, when the dependent variable is only equal to 0 (or 1 for Logit) for one cluster value this leads to a perfect fit for that cluster value by setting its associated cluster coefficient to `-Inf`. Thus these observations need to be removed before estimation. This function gives the observations to be removed. Not that by default the function `femlm` drops them before performing the estimation.

Usage

```
obs2remove(fml, data, family = c("poisson", "negbin", "logit"))
```

Arguments

`fml`	A formula contaning the dependent variable and the clusters. It can be of the type: `y ~ cluster_1 + cluster_2` or `y ~ x1	cluster_1 + cluster_1` (in which case variables before the pipe are ignored).
`data`	A data.frame containing the variables in the formula.
`family`	Character scalar: either "poisson" (default), "negbin" or "logit".

Value

It returns an integer vector of observations to be removed. If no observations are to be removed, an empty integer vector is returned. In both cases, it is of class `femlm.obs2remove`. The vector has an attribut `cluster` which is a list giving the IDs of the clusters that have been removed, for each cluster dimension.

Examples

```
base = iris
# v6: Petal.Length with only 0 values for 'setosa'
base$v6 = base$Petal.Length
base$v6[base$Species == "setosa"] = 0

(x = obs2remove(v6 ~ Species, base))
attr(x, "cluster")

# The two results are identical:
res_1 = femlm(v6 ~ Petal.Width | Species, base)
# => warning + obsRemoved is created

res_2 = femlm(v6 ~ Petal.Width | Species, base[-x, ])
# => no warning because observations are removed before

res2table(res_1, res_2)

all(res_1$obsRemoved == x)
```

`plot.femlm.allClusters` *Displaying the most notable fixed-effects*

Description

This function plots the 5 fixed-effects with the highest and lowest values, for each of the clusters. It takes as an argument the fixed-effects obtained from the function `getFE` after and estimation using `femlm`.

Usage

```
## S3 method for class 'femlm.allClusters'
plot(x, n = 5, ...)
```

Arguments

`x`	An object obtained from the function `getFE`.
`n`	The number of fixed-effects to be drawn. Defaults to 5.
`...`	Not currently used.
	Note that the fixed-effect coefficients might NOT be interpretable. This function is useful only for fully regular panels.
	If the data are not regular in the cluster coefficients, this means that several 'reference points' are set to obtain the fixed-effects, thereby impeding their interpretation. In this case a warning is raised.

Author(s)

Laurent Berge

See Also

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

Examples

```
data(trade)

# We estimate the effect of distance on trade
# => we account for 3 cluster effects
est_pois = femlm(Euros ~ log(dist_km)|Origin+Destination+Product, trade)

# obtaining the cluster coefficients
fe_trade = getFE(est_pois)

# plotting them
plot(fe_trade)
```

---

predict.femlm *Predict method for femlm fits*

Description

This function obtains prediction from a fitted model estimated with femlm.

Usage

```
## S3 method for class 'femlm'
predict(object, newdata, type = c("response", "link"),
  ...)
```

Arguments

object	An object of class femlm. Typically the result of a femlm estimation.
newdata	A data.frame containing the variables used to make the prediction. If not provided, the fitted expected (or linear if type = "link") predictors are returned.
type	Character either equal to "response" (default) or "link". If type="response", then the output is at the level of the response variable, i.e. it is the expected predictor $E(Y\|X)$. If "link", then the output is at the level of the explanatory variables, i.e. the linear predictor $X \cdot \beta$.
...	Not currently used.

Value

It returns a numeric vector of length equal to the number of observations in argument `newdata`.

Author(s)

Laurent Berge

See Also

`femlm`, `update.femlm`, `summary.femlm`, `vcov.femlm`, `getFE`.

Examples

```
# Estimation on iris data
res = femlm(Sepal.Length ~ Petal.Length | Species, iris)

# what would be the prediction if the data was all setosa?
newdata = data.frame(Petal.Length = iris$Petal.Length, Species = "setosa")
pred_setosa = predict(res, newdata = newdata)

# Let's look at it graphically
plot(c(1, 7), c(3, 11), type = "n", xlab = "Petal.Length",
     ylab = "Sepal.Length")

newdata = iris[order(iris$Petal.Length), ]
newdata$Species = "setosa"
lines(newdata$Petal.Length, predict(res, newdata))

# versicolor
newdata$Species = "versicolor"
lines(newdata$Petal.Length, predict(res, newdata), col=2)

# virginica
newdata$Species = "virginica"
lines(newdata$Petal.Length, predict(res, newdata), col=3)

# The original data
points(iris$Petal.Length, iris$Sepal.Length, col = iris$Species, pch = 18)
legend("topleft", lty = 1, col = 1:3, legend = levels(iris$Species))
```

---

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, n, ...)
```

Arguments

x	A femlm object. Obtained using femlm.
n	Integer, number of coefficients to display. By default, all estimated coefficients are displayed.
...	Other arguments to be passed to vcov.femlm.

Author(s)

Laurent Berge

See Also

See also the main estimation functions 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

```
# Load trade data
data(trade)

# We estimate the effect of distance on trade => we account for 3 cluster effects
est_pois = femlm(Euros ~ log(dist_km)|Origin+Destination+Product, trade)

# displaying the results
print(est_pois)

# with other type of standard error:
print(est_pois, se = "c")
```

---

print.femlm.obs2remove

Print method for femlm.obs2remove objects

Description

This function show synthetizes the information of function obs2remove. It reports the number of observations to be removed as well as the number of clusters removed per cluster dimension.

Usage

```
## S3 method for class 'femlm.obs2remove'
print(x, ...)
```

Arguments

x	A femlm.obs2remove object obtained from function obs2remove.
...	Not currently used.

Examples

```
base = iris
# v6: Petal.Length with only 0 values for 'setosa'
base$v6 = base$Petal.Length
base$v6[base$Species == "setosa"] = 0

(x = obs2remove(v6 ~ Species, base))
attr(x, "cluster")
```

---

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, depvar, drop, order, digits = 4,
  pseudo = TRUE, convergence, signifCode = c(`***` = 0.01, `**` = 0.05,
  `*` = 0.1), subtitles, keepFactors = FALSE, family)
```

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 computed: "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. For one-way clustering, this argument can directly be a vector (instead of a list). If the estimation has been done

with cluster variables, you can give a character vector of the dimensions over which to cluster the SE.

`depvar` Logical, default is missing. Whether a first line containing the dependent variables should be shown. By default, the dependent variables are shown only if they differ across models.

`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.

`digits` Integer, default is 4. The number of digits to be displayed.

`pseudo` Logical, default is `TRUE`. Should the pseudo R2 be displayed?

`convergence` Logical, default is missing. Should the convergence state of the algorithm be displayed? By default, convergence information is displayed if at least one model did not converge.

`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.

`family` A logical, default is missing. Whether to display the families of the models. By default this line is displayed when at least two models are from different families.

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

```
# two fitted models with different expl. variables:
res1 = femlm(Sepal.Length ~ Sepal.Width + Petal.Length +
            Petal.Width | Species, iris)
# estimation without clusters
res2 = update(res1, . ~ Sepal.Width | 0)

# We export the two results in one Latex table:
res2table(res1, res2)

# With clustered standard-errors + showing the dependent variable
res2table(res1, res2, se = "cluster", cluster = iris$Species, depvar = TRUE)

# Changing the model names + the order of the variables
# + dropping the intercept.
res2table(model_1 = res1, res2,
          order = c("Width", "Petal"), drop = "Int",
          signifCode = c("**" = 0, "*" = 0.2, "n.s."=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, replace = FALSE,
  convergence, 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, family,
  powerBelow = -5)
```

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 computed: "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`. For one-way clustering, this argument can directly be a vector (instead of a list). If the estimation has been done with cluster variables, you can give a character vector of the dimensions over which to cluster the SE.

`digits` Integer, default is 4. 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`.

`replace` Logical, default is `FALSE`. Only used if option `file` is used. Should the Latex table be written in a new file that replaces any existing file?

`convergence` Logical, default is missing. Should the convergence state of the algorithm be displayed? By default, convergence information is displayed if at least one model did not converge.

`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.
family	A logical, default is missing. Whether to display the families of the models. By default this line is displayed when at least two models are from different families.
powerBelow	Integer, default is -5. A coefficient whose value is below 10**(powerBelow+1) is written with a power in Latex. For example 0.0000456 would be written 4.56$\times 10^{-5}$ by default. Setting powerBelow = -6 would lead to 0.00004 in Latex.

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

```
# two fitted models with different expl. variables:
res1 = femlm(Sepal.Length ~ Sepal.Width + Petal.Length +
            Petal.Width | Species, iris)
res2 = femlm(Sepal.Length ~ Petal.Width | Species, iris)

# We export the three results in one Latex table,
# with clustered standard-errors:
res2tex(res1, res2, se = "cluster")

# Changing the names & significance codes
res2tex(res1, res2, dict = c(Sepal.Length = "The sepal length", Sepal.Width = "SW"),
        signifCode = c("**" = 0.1, "*" = 0.2, "n.s."=1))
```

`resid.femlm` *Extracts residuals from a femlm object*

Description

This function extracts residuals from a fitted model estimated with `femlm`.

Usage

```
## S3 method for class 'femlm'
resid(object, ...)

## S3 method for class 'femlm'
residuals(object, ...)
```

Arguments

`object`	An object of class `femlm`. Typically the result of a `femlm` estimation.
`...`	Not currently used.

Details

The residuals returned are the difference between the dependent variable and the expected predictor.

Value

It returns a numeric vector of the length the number of observations used for the estimation.

Author(s)

Laurent Berge

See Also

`femlm`, `fitted.femlm`, `predict.femlm`, `summary.femlm`, `vcov.femlm`, `getFE`.

Examples

```
# simple estimation on iris data, clustering by "Species"
res_poisson = femlm(Sepal.Length ~ Sepal.Width + Petal.Length +
                    Petal.Width | Species, iris)

# we plot the residuals
plot(resid(res_poisson))
```

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 computed: "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. For one-way clustering, this argument can directly be a vector (instead of a list). If the estimation has been done with cluster variables, you can give a character vector of the dimensions over which to cluster the SE.
dof_correction	Logical, default is FALSE. Should there be a degree of freedom correction to the standard errors of the coefficients?
forceCovariance	(Advanced users.) 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	(Advanced users.) 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).

se	The new standard-errors (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

```
# Load trade data
data(trade)

# We estimate the effect of distance on trade (with 3 cluster effects)
est_pois = femlm(Euros ~ log(dist_km)|Origin+Destination+Product, trade)

# 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)

# Alternative ways to cluster the SE:
## Not run:
# two-way clustering: Destination and Product
summary(est_pois, se = "twoway", cluster = c("Destination", "Product"))
summary(est_pois, se = "twoway", cluster = list(trade$Destination, trade$Product))

## End(Not run)
```

---

summary.femlm.allClusters

Summary method for cluster coefficients

Description

This function summarizes the main characteristics of the cluster coefficients. It shows the number of fixed-effects that have been set as references and the first elements of the fixed-effects.

Usage

```
## S3 method for class 'femlm.allClusters'
summary(object, n = 5, ...)
```

Arguments

object	An object returned by the function getFE.
n	Positive integer, defaults to 5. The n first fixed-effects for each cluster are reported.
...	Not currently used.

Value

It prints the number of fixed-effect coefficients per cluster, as well as the number of fixed-effects used as references for each cluster, and the mean and variance of the cluster coefficients. Finally it reports the first 5 elements of each cluster.

Author(s)

Laurent Berge

See Also

femlm, getFE, plot.femlm.allClusters.

Examples

```
data(trade)

# We estimate the effect of distance on trade
# => we account for 3 cluster effects
est_pois = femlm(Euros ~ log(dist_km)|Origin+Destination+Product, trade)

# obtaining the cluster coefficients
fe_trade = getFE(est_pois)

# printing some summary information on the cluster coefficients:
fe_trade
```

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`.

- Origin: 2-digits codes of the countries of origin of the trade flow.
- Destination: 2-digits codes of the countries of destination of the trade flow.
- Products: Number representing the product categories (from 1 to 20).
- Year: Years from 2007 to 2016
- dist_km: Geographic distance in km between the centers of the countries of origin and destination.
- Euros: The total amount of trade flow in million euros for the specific year/product category/origin-destination country pair.

Source

This data has been extrated from Eurostat on October 2017.

update.femlm *Updates a femlm estimation*

Description

Updates and re-estimates a `femlm` model. This function updates the formulas and use previous starting values to estimate a new `femlm` model. The data is obtained from the original `call`.

Usage

```
## S3 method for class 'femlm'
update(object, fml.update, ...)
```

Arguments

object	An object of class femlm. Typically the result of a femlm estimation.
fml.update	Changes to be made to the original argument fml. See more information on update.formula. You can add/withdraw both variables and clusters. E.g. . ~ . + x2 | . + z2 would add the variable x2 and the cluster z2 to the former estimation.
...	Other arguments to be passed to the function femlm.

Value

It returns a femlm object (see details in femlm.

Author(s)

Laurent Berge

See Also

femlm, predict.femlm, summary.femlm, vcov.femlm, getFE.

Examples

```
# Example using trade data
data(trade)

# main estimation
est_pois <- femlm(Euros ~ log(dist_km) | Origin + Destination, trade)

# we add the variable log(Year)
est_2 <- update(est_pois, . ~ . + log(Year))

# we add another cluster: "Product"
est_3 <- update(est_2, . ~ . | . + Product)

# we remove the cluster "Origin" and the variable log(dist_km)
est_4 <- update(est_3, . ~ . - log(dist_km) | . - Origin)

# Quick look at the 4 estimations
res2table(est_pois, est_2, est_3, est_4)
```

---

vcov.femlm *Extract the variance/covariance of a femlm fit*

Description

This function extracts the variance-covariance of estimated parameters from a model estimated with femlm.

Usage

```
## S3 method for class 'femlm'
vcov(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 computed: "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. For one-way clustering, this argument can directly be a vector (instead of a list). If the estimation has been done with cluster variables, you can give a character vector of the dimensions over which to cluster the SE.
dof_correction	Logical, default is FALSE. Should there be a degree of freedom correction to the standard errors of the coefficients?
forceCovariance	(Advanced users.) 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	(Advanced users.) 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).
...	Other arguments to be passed to summary.femlm. The computation of the VCOV matrix is first done in summary.femlm.

Value

It returns a $N \times N$ square matrix where N is the number of variables of the fitted model. This matrix has an attribute "type" specifying how this variance/covariance matrix has been commputed (i.e. was it created using White correction, or was it clustered along a specific factor, etc).

Author(s)

Laurent Berge

See Also

femlm, summary.femlm, confint.femlm, resid.femlm, predict.femlm, getFE.

Examples

```
# Load trade data
data(trade)

# We estimate the effect of distance on trade (with 3 fixed-effects)
est_pois = femlm(Euros ~ log(dist_km) + log(Year) | Origin + Destination +
                 Product, trade)

# "normal" VCOV
vcov(est_pois)

# "white" VCOV
vcov(est_pois, se = "white")

# "clustered" VCOV (with respect to the Origin factor)
vcov(est_pois, se = "cluster")

# "clustered" VCOV (with respect to the Product factor)
vcov(est_pois, se = "cluster", cluster = trade$Product)
# another way to make the same request:
vcov(est_pois, se = "cluster", cluster = "Product")

# Another estimation without cluster:
est_pois_simple = femlm(Euros ~ log(dist_km) + log(Year), trade)

# We can still get the clustered VCOV,
# but we need to give the cluster-vector:
vcov(est_pois_simple, se = "cluster", cluster = trade$Product)
```

Index