Package 'FENmlm'

October 3, 2015

Type Package

Title Fixed Effects Nonlinear Maximum Likelihood Models

Version 1.0

Date 2015-10-02

Author Laurent Berge

Maintainer Laurent Berge <laurent.berge@u-bordeaux.fr>

Depends stats, numDeriv, Matrix, MASS

Description Efficient estimation of fixed-effect maximum likelihood models with, possibly, non-linear right hand sides.

License GPL (>= 2)

LazyData TRUE

NeedsCompilation no

Repository CRAN

Date/Publication 2015-10-03 19:21:39

R topics documented:

FENmlm-package *Fixed Effects Nonlinear Maximum Likelihood Models*

Description

Efficient estimation of fixed-effect maximum likelihood models with, possibly, non-linear right hand sides.

Details

Package:	FENmlm
Type:	Package
Version:	1.0
Date:	2015-10-02
License:	GPL-2
LazyLoad:	yes

This package intends to efficiently estimate fixed-effect maximum likelihood models. The function *feNmlm* perfoms estimates fixed-effect maximum likelihood models with non-linear right hand sides. The function *femlm* is similar but is restricted to linear right hand sides. Several features are also included such as the possibility to easily compute different types of standard-errors (including one-way and two-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 u-bordeaux.fr>

---

femlm *Fixed effects maximum likelihood models.*

Description

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

Usage

```
femlm(linear.fml, data, dummy, linear.start = 0,
      useHessian = TRUE, opt_method = c("nlminb", "optim"),
      debug = FALSE, family = c("poisson", "negbin", "logit"),
      opt.control=list(),optim.method="BFGS",...)
```

Arguments

linear.fml	A formula. The linear formula to be estimated.
family	Character scalar. It should provide the family. Currently family="poisson", family="negbin" and family="logit" are implemented. Note that the log link is used by default.
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.
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, use start.init. Yet this is not recommended.
dummy	Character vector. The name/s of a/some variable/s within the dataset. These variables should contain the identifier of each observation (e.g., think of it as a panel identifier).
linear.start	Numeric named vector. The starting values of the linear part. If it is just a numeric scalar, all coefficients are set to linear.start.
useHessian	Logical. (Only if omptimization method is optim). Should the Hessian be computed in the optimization stage? Default is TRUE.
opt_method	Character scalar. Which optimization method should be used. Either nlminb or optim. Default is nlminb.
opt.control	List of elements to be passed to the optimization method (nlminb or optim).
optim.method	Character scalar. If opt_method="optim", it is the algorithm to be used by optim (default is "BFGS"). See optim help pages for detail.
debug	Logical. If TRUE then the log-likelihood as well as all parameters are printed at each iteration. Default is FALSE.
...	Not currently used.

Value

An feNmlm 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.
nonlinear.fml	The nonlinear formula of the call. It also contains the dependent variable.
linear.formula	The linear formula of the call.
ll_null	Log-likelyhood of the null model

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.
expected.predictor	The expected predictor is the expected value of the dependent variable.
cov.unscaled	The variance covariance matrix of the parameters.
sd	The standard error of the parameters.

Author(s)

Laurent Berge

See Also

See also feNmlm.

Examples

```
#The data
n = 100
x = rnorm(n,1,5)**2
y = rnorm(n,-1,5)**2
z = rpois(n,x*y)
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
est_poisson
est_negbin

# Changing the way the standard errors are computed:
summary(est_poisson,sd="white")
summary(est_negbin,sd="white")

#
# Now with dummies
#

# 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 dummy
est_poisson = femlm(z~log(x)+log(y),base,family="poisson",dummy=c("id1","id2"))
# Displaying the results with twoway clustered santard-errors
print(est_poisson,"t")
```

`feNmlm` *Fixed effects non-linear maximum likelihood models*

Description

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

Usage

```
feNmlm(fml,data,linear.fml,start,lower,upper,
       env,dummy,start.init,nl.gradient,linear.start=0,
       jacobian.method=c("simple","Richardson"),useHessian=TRUE,
       d.hessian,opt_method=c("nlminb","optim"),debug=FALSE,
       family=c("poisson","negbin","logit"),
       opt.control=list(),optim.method="BFGS",...)
```

Arguments

`fml`	A formula. This formula must provide the dependent variable as well as the non linear part of the right hand side (RHS). It can be for instance `y~a*log(b*x + c*x^3)`. If there is no non-linear part, the RHS of the formula should be 0: e.g. `y~0`.
`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.
`family`	Character scalar. It should provide the family. Currently `family="poisson"`, `family="negbin"` and `family="logit"` are implemented. Note that the log link is used by default.
`linear.fml`	A formula with no left hand side. This formula states the linear parameters (as the constant for instance). Putting linear parameters in this formula enhances A LOT the performance of the algorithm. Example: `linear.fml = ~ 1` to include only the constant, or `linear.fml = ~ z + factor(f)` for other variables along with the constant. Note that by default there is not any linear parameter (not even the constant).

`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, use `start.init`. Yet this is not recommended.

`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, problems can be raised when computing the Jacobian or the Hessian. A proper setting of `lower` or by using `d.hessian` or `d.jacobian` can solve these issues. See details.

`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, problems can be raised when computing the Jacobian or the Hessian. A proper setting of `upper` or a proper use of `d.hessian` can solve this issue. See details.

`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.)

`dummy` Character vector. The name/s of a/some variable/s within the dataset. These variables should contain the identifier of each observation (e.g., think of it as a panel identifier).

`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`.

`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.

`jacobian.method`
Character scalar. Provides the method used to numerically compute the jacobian. Can be either `"simple"` or `"Richardson"`. Default is `"simple"`. See the help of numDeriv for more information.

`useHessian` Logical. (Only if omptimization method is `optim`). Should the Hessian be computed in the optimization stage? Default is `TRUE`.

`d.hessian` Numeric scalar. It provides an argument to the function `hessian` of the package `numDeriv`. It defines the step used to compute the hessian. The default being 0.1, it can lead to problems when some parameters are at their lower or upper bound. See details for more information.

`opt_method` Character scalar. Which optimization method should be used. Either `nlminb` or `optim`. Default is `nlminb`.

`opt.control` List of elements to be passed to the optimization method (nlminb or optim).

`optim.method` Character scalar. If `opt_method="optim"`, it is the algorithm to be used by optim (default is "BFGS"). See `optim` help pages for detail.

debug	Logical. If TRUE then the log-likelihood as well as all parameters are printed at each iteration. Default is FALSE.
...	Not currently used.

Details

When the paramters are at their lower or upper bound, there can be problems when computing the Hessian. This is because the values of the paremeters are shifted to compute numerically the hessian. The defaults of those steps are 0.1 (see the help pages of hessian). Thus, in the case where the non-linear part CANNOT be estimated when the parameter is beyond its bound, the hessian will not be possibly computed numerically. Thus the most straightforward way to circumvent this problem is to either rise the lower (resp. lower the upper) bound by more than 0.1, or to set d.hessian to a lower value (while slighlty rising/lowering the bound).

Value

An feNmlm 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.
nonlinear.fml	The nonlinear formula of the call. It also contains the dependent variable.
linear.formula	The linear formula of the call.
ll_null	Log-likelyhood of the null model
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.
expected.predictor	The expected predictor is the expected value of the dependent variable.
cov.unscaled	The variance covariance matrix of the parameters.
sd	The standard error of the parameters.

Author(s)

Laurent Berge

Examples

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

#Comparing the results of a 'linear' function
est0L = feNmlm(z~0,base,~log(x)+log(y),family="poi")
est0NL = feNmlm(z~a*log(x)+b*log(y),base,start = list(a=0,b=0),
                family="poisson", linear.fml=~1)

est0NL_hess = feNmlm(z~a*log(x)+b*log(y),base,start = list(a=0,b=0),
                     family="poisson", linear.fml=~1, useHessian=TRUE)

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

#Using a non-linear form
est1L = feNmlm(z2~0,base,~log(x)+log(y),family="poi")
est1NL = feNmlm(z2~log(a*x + b*y),base,start = list(a=1,b=2),family="poisson")
est1NL_hess = feNmlm(z2~log(a*x + b*y),base,start = list(a=1,b=2),
                     family="poisson",useHessian=TRUE)

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

est1NL_grad = feNmlm(z2~log(a*x + b*y), base, start = list(a=1,b=2),
                     family="poisson", nl.gradient = ~myGrad(a,x,b,y))
```

---

getFE *Extract the Fixed-Effects from a* feNmlm *estimation.*

Description

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

Usage

```
getFE(x)
```

Arguments

x	A feMmlm object.

Value

A list containig the vectors of the fixed effects.

Author(s)

Laurent Berge

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)
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 dummy
est_poisson = femlm(z~log(x)+log(y),base,family="poisson",dummy=c("id1","id2"))

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

---

print.feNmlm *A print facility for* feNmlm *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 'feNmlm'
print(x, sd = c("standard", "white","cluster","twoway"),cluster, ...)
```

Arguments

x	A feNmlm object.
sd	Character scalar. Which kind of standard error should be prompted: "standard" (default), "White", or "cluster"?
cluster	A list of vectors. Used only if `sd = "cluster"` or `sd="twoway"`. The vectors should give the cluster of each observation. Note that if the estimation was run using `dummy`, the standard error is automatically clustered along the cluster given in `feNmlm`.
...	Currently unused.

Author(s)

Laurent Berge

See Also

See also `feNmlm`.

Examples

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

#Comparing the results of a 'linear' function
est0L = feNmlm(z~0,base,~log(x)+log(y),family="poi")
est0NL = feNmlm(z~a*log(x)+b*log(y),base,start = list(a=0,b=0),
family="poisson", linear.fml=~1)

print(est0L)
print(est0NL)

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

#Using a non-linear form
est1L = feNmlm(z2~0,base,~log(x)+log(y),family="poi")
est1NL = feNmlm(z2~log(a*x + b*y),base,start = list(a=1,b=2),family="poisson")
```

res2table *Facility to display the results of multiple* feNmlm *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(..., sd = c("standard", "white", "cluster", "twoway"),
          cluster, digits = 4, pseudo = TRUE,
          sdBelow = TRUE, drop, order, convergence = TRUE)
```

Arguments

...	Used to capture different feNmlm objects. Note that any other type of element is discarded.
sd	A character scalar. The standard-error is computed similarly for each feNmlm object. Which kind of standard error should be prompted: "standard" (default), "White", "cluster" or "twoway"?
cluster	A list of vectors. Used only if sd = "cluster" or sd="twoway". The vectors should give the cluster of each observation. Note that if the estimation was run using dummy, the standard error is automatically clustered along the cluster given in feNmlm.
digits	Integer. The number of digits to be displayed.
pseudo	Logical. Should the pseudo R2 be displayed? (Default is TRUE.)
sdBelow	Logical. Should the standard-error be displayed below the coefficients? (Default is TRUE.)
drop	Character vector. This element is used if some variables are not to be displayed. This should be a regular expression (see regexp 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 regexp 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. Should the convergence state of the algorithm be displayed? (Default is TRUE.)

Value

There is nothing returned, the result is only displayed on the console.

Author(s)

Laurent Berge

Examples

```
n = 100
x = rnorm(n,1,5)**2
y = rnorm(n,-1,5)**2
z = rpois(n,x*y)
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 display the two results in one table:
res2table(est_poisson,est_negbin)
```

---

res2tex *Facility to export the results of multiple* feNmlm *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(..., sd = c("standard", "white", "cluster", "twoway"),
        cluster, digits = 4, pseudo = TRUE,
        title, sdBelow = TRUE, drop, order, dict, file, append = TRUE,
        convergence = TRUE)
```

Arguments

...	Used to capture different feNmlm objects. Note that any other type of element is discarded.
sd	A character scalar. The standard-error is computed similarly for each feNmlm object. Which kind of standard error should be prompted: "standard" (default), "White", "cluster" or "twoway"?
cluster	A list of vectors. Used only if sd = "cluster" or sd="twoway". The vectors should give the cluster of each observation. Note that if the estimation was run using dummy, the standard error is automatically clustered along the cluster given in feNmlm.

digits	Integer. The number of digits to be displayed.
pseudo	Logical. Should the pseudo R2 be displayed? (Default is TRUE.)
title	Character scalar. The title of the Latex table.
sdBelow	Logical. Should the standard-error be displayed below the coefficients? (Default is TRUE.)
drop	Character vector. This element is used if some variables are not to be displayed. This should be a regular expression (see regexp 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 regexp 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. Default is missing. 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. Default is missing. If provided, the Latex table will be saved in a file whose path is file.
append	Logical. Only used if option file is used. Should the Latex table be appended to the existing file? (Default is TRUE.)
convergence	Logical. Should the convergence state of the algorithm be displayed? (Default is TRUE.)

Value

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

Author(s)

Laurent Berge

Examples

```
n = 100
x = rnorm(n,1,5)**2
y = rnorm(n,-1,5)**2
z = rpois(n,x*y)
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)
```

---

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

Description

This function is similar to print.feNmlm. 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 'feNmlm'
summary(object,sd=c("standard","white","cluster","twoway"),
         cluster,dof_correction=TRUE,...)
```

Arguments

object	A feNmlm object.
sd	Character scalar. Which kind of standard error should be prompted: "standard" (default), "White", or "cluster"?
cluster	A list of vectors. Used only if sd = "cluster" or sd="twoway". The vectors should give the cluster of each observation. Note that if the estimation was run using dummy, the standard error is automatically clustered along the cluster given in feNmlm.
dof_correction	Logical. Should a finite sample correcton be applied? (Default is TRUE.)
...	Currently unused.

Value

The same values as a feMmlm object plus:

vcov	The variance-covariance matrix whose type is the one requested by the user.

Author(s)

Laurent Berge

Examples

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

#Comparing the results of a 'linear' function
est0L = feNmlm(z~0,base,~log(x)+log(y),family="poi")
est0NL = feNmlm(z~a*log(x)+b*log(y),base,start = list(a=0,b=0),
family="poisson",linear.fml=~1)

# Displaying the summary
summary(est0L,sd="white")
myWhiteVcov = summary(est0L,sd="white")$vcov
```

Index