Package ‘geeM’

May 8, 2026

Type Package

Title Solve Generalized Estimating Equations

Version 0.10.1

Date 2018-05-21

Author Lee McDaniel [aut, cre],
Nick Henderson [aut],
Melanie Prague [ctb] (Suggested code to fix weighting)

Maintainer Lee McDaniel <lmcda4@lsuhsc.edu>

Depends Matrix

Imports stats, methods

Suggests geepack, testthat, MuMIn

Description GEE estimation of the parameters in mean structures with possible correlation between the outcomes. User-specified mean link and variance functions are allowed, along with observation weighting. The 'M' in the name 'geeM' is meant to emphasize the use of the Matrix package, which allows for an implementation based fully in R.

License GPL-3

ByteCompile TRUE

NeedsCompilation no

Repository CRAN

Date/Publication 2018-06-18 15:49:55 UTC

Contents

geem 2

Index **6**

geem *Fit Generalized Estimating Equations*

Description

Calculate coefficients and nuisance parameters using generalized estimating equations. Link and Variance functions can be specified by the user. Similar to glm.

Usage

```
geem(formula, id, waves=NULL, data = parent.frame(), family = gaussian,
 corstr = "independence", Mv = 1, weights = NULL, corr.mat = NULL, init.beta = NULL,
 init.alpha = NULL, init.phi = 1, scale.fix = FALSE, nodummy=FALSE, sandwich = TRUE,
 useP = TRUE, maxit = 20, tol = 1e-05)
```

Arguments

`formula` a formula expression similar to that for glm, of the form `response~predictors`. An offset is allowed, as in `glm`.

`id` a vector identifying the clusters. By default, data are assumed to be sorted such that observations in a cluster are in consecutive rows and higher numbered rows in a cluster are assumed to be later. If NULL, then each observation is assigned its own cluster.

`waves` an integer vector identifying components of a cluster. For example, this could be a time ordering. If integers are skipped within a cluster, then dummy rows with weight 0 are added in an attempt to preserve the correlation structure (except if `corstr = "exchangeable"` or `"independent"`). This can be skipped by setting `nodummy=TRUE`.

`data` an optional data frame containing the variables in the model.

`family` will determine the link and variance functions. The argument can be one of three options: a `family` object, a character string, or a list of functions.

For more information on how to use `family` objects, see `family`.

If the supplied argument is a character string, then the string should correspond to one of the family objects.

In order to define a link function, a list must be created with the components `(LinkFun, VarFun, InvLink, InvLinkDeriv)`, all of which are vectorized functions. If the components in the list are not named as

`(LinkFun, VarFun, InvLink, InvLinkDeriv)`, then `geem` assumes that the functions are given in that order. LinkFun and VarFun are the link and variance functions. InvLink and InvLinkDeriv are the inverse of the link function and the derivative of the inverse of the link function and so are decided by the choice of the link function.

`corstr` a character string specifying the correlation structure. Allowed structures are: `"independence"`, `"exchangeable"`, `"ar1"`, `"m-dependent"`, `"unstructured"`, `"fixed"`, and `"userdefined"`. Any unique substring may be supplied. If

"fixed" or "userdefined", then corr.mat must be specified. If "m-dependent", then Mv is relevant.

Mv — for "m-dependent", the value for m.

weights — A vector of weights for each observation. If an observation has weight 0, it is excluded from the calculations of any parameters. Observations with a NA anywhere (even in variables not included in the model) will be assigned a weight of 0. Note that these weights are now the same as PROC GEE weights and not PROC GENMOD.

corr.mat — the correlation matrix for "fixed". Matrix should be symmetric with dimensions >= the maximum cluster size. If the correlation structure is "userdefined", then this is a matrix describing which correlations are the same.

init.beta — an optional vector with the initial values of beta. If not specified, then the intercept will be set to InvLink(mean(response)). init.beta must be specified if not using an intercept.

init.alpha — an optional scalar or vector giving the initial values for the correlation. If provided along with Mv>1 or unstructured correlation, then the user must ensure that the vector is of the appropriate length.

init.phi — an optional initial overdispersion parameter. If not supplied, initialized to 1.

scale.fix — if set to TRUE, then the scale parameter is fixed at the value of init.phi.

nodummy — if set to TRUE, then dummy rows will not be added based on the values in waves.

sandwich — if TRUE, calculate robust variance.

useP — if set to FALSE, do not use the n-p correction for dispersion and correlation estimates, as in Liang and Zeger. This can be useful when the number of observations is small, as subtracting p may yield correlations greater than 1.

maxit — maximum number of iterations.

tol — tolerance in calculation of coefficients.

Details

Users may specify functions for link and variance functions, but the functions must be vectorized functions. See Vectorize for an easy way to vectorize functions. Vectorize should be used sparingly, however, as it can lead to fairly slow function calls. Care must be taken to ensure that convergence is possible with non-standard functions.

Offsets must be specified in the model formula, as in glm.

For the "userdefined" correlation option, the function accepts a matrix with consecutive integers. geem only looks at the upper triangle of the matrix. Any entry given as 0 will be fixed at 0. All entries given as 1 will be assumed to be the same as each other and will be assumed to be possibly different from entries with a 2, and so on.

If observations are dropped because they have a weight of 0, then the denominator for the moment estimates of the correlation matrices are calculated using the number of non-zero Pearson residuals for the correlation structures unstructured, userdefined and m-dependent with Mv>1. Therefore residuals numerically equal to 0 may cause problems in the calculation of correlation parameters.

Value

An object of class "geem" representing the fit.

Author(s)

Lee McDaniel and Nick Henderson

See Also

glm, formula, family

Examples

```
### Generated Negative Binomial Data
generatedata <- function(beta,alpha,gamma,X,T,n)  {
     mean.vec <- exp(crossprod(t(X),beta))
     y <- matrix(0,nrow=n,ncol=T)
     y[,1] <- rnbinom(n,mu = mean.vec[1],size=mean.vec[1]/gamma)
     for (i in 1:n)  {
        for (t in 2:T)  {
           innovation.mean <- mean.vec[t] - alpha*(sqrt(mean.vec[t]*mean.vec[t-1]))
           I <- rnbinom(1,mu= innovation.mean,size= innovation.mean/gamma)
           first.shape <- alpha*sqrt(mean.vec[t]*mean.vec[t-1])/gamma
           second.shape <- mean.vec[t-1]/gamma - first.shape
           u <- rbeta(1,shape1 = first.shape,shape2=second.shape)
           a <- rbinom(1,size=y[i,t-1],prob=u)
           y[i,t] = a + I
        }
     }
     longform <- c(t(y))
     print(apply(y,2,mean))
    simdata <- data.frame(count = longform, time = rep(X[,2],times=n),subject=rep(c(1:n),each=T))
     return(simdata)
}

X <- cbind(rep(1,5),c(-.5,-.25,0,.25,.5))
testdat <- generatedata(beta=c(1,.5),alpha=.2,gamma=.5,X=X,T=5,n=3000)
far1 <- geem(count~ time, id=subject ,data = testdat, family=poisson,
corstr="ar1")

### Ohio respiratory data from geepack
if(require(geepack)){
data("ohio", package="geepack")
resplogit <- geem(resp ~ age + smoke + age:smoke, id=id, data = ohio, family = binomial,
corstr = "m-dep" , Mv=1)

  LinkFun <- function(arg){qcauchy(arg)}
  InvLink <- function(arg){pcauchy(arg)}
  InvLinkDeriv <- function(arg){dcauchy(arg)}
  VarFun <- function(arg){arg*(1-arg)}
  FunList <- list(LinkFun, VarFun, InvLink, InvLinkDeriv)
```

```
  respcauchit <- geem(resp ~ age + smoke + age:smoke, id=id, data = ohio, family = FunList,
   corstr = "m-dep" , Mv=1)
}

### Seizure data from geepack
if(require(geepack)){
data("seizure", package="geepack")
seiz.l <- reshape(seizure,
varying=list(c("base","y1", "y2", "y3", "y4")),
v.names="y", times=0:4, direction="long")
seiz.l <- seiz.l[order(seiz.l$id, seiz.l$time),]
seiz.l$t <- ifelse(seiz.l$time == 0, 8, 2)
seiz.l$x <- ifelse(seiz.l$time == 0, 0, 1)

seiz <- geem(y~ x + trt + x:trt+ offset(log(t)), id=id,data = seiz.l,
family = poisson, corstr = "exchangeable")
}
```

Index

∗ **models**
geem, 2
∗ **robust**
geem, 2

family, *2*, *4*
formula, *4*

geeM *(geem)*, 2
geem, 2
geeM-package *(geem)*, 2
glm, *2*, *4*

Vectorize, *3*