Package ‘geeasy’

July 23, 2025

Type Package

Title Solve Generalized Estimating Equations for Clustered Data

Version 0.1.3

Maintainer Søren Højsgaard <sorenh@math.aau.dk>

Depends geepack, stats

Imports ggplot2, geeM, lme4, methods, Matrix, MESS

Suggests testthat, MuMIn

Description Estimation of generalized linear models with correlated/clustered observations by use of generalized estimating equations (GEE). See e.g. Halekoh and Højsgaard, (2005, <doi:10.18637/jss.v015.i02>), for details. Several types of clustering are supported, including exchangeable variance structures, AR1 structures, M-dependent, user-specified variance structures and more. The model fitting computations are performed using modified code from the 'geeM' package, while the interface and output objects have been written to resemble the 'geepack' package. The package also contains additional tools for working with and inspecting results from the 'geepack' package, e.g. a 'confint' method for 'geeglm' objects from 'geepack'.

License GPL-3

RoxygenNote 7.3.2

Encoding UTF-8

Language en-US

NeedsCompilation no

Author Anne Helby Petersen [aut],
Lee McDaniel [aut] (Author of geeM),
Claus Ekstrøm [ctb] (Wrote code for drop1 methods),
Søren Højsgaard [aut, cre] (Author of geepack)

Repository CRAN

Date/Publication 2025-07-23 07:50:09 UTC

Contents

`confint.geelm` *Confidence Intervals for geelm objects*

Description

Compute Wald confidence intervals for mean structure parameters of geelm object.

Usage

```
## S3 method for class 'geelm'
confint(object, parm = NULL, level = 0.95, std.err = "san.se", ...)

## S3 method for class 'geeglm'
confint(object, parm = NULL, level = 0.95, std.err = "san.se", ...)
```

Arguments

`object`	a fitted model object.
`parm`	specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
`level`	the confidence level required.
`std.err`	Which standard error estimation method that should be used for computing the confidence intervals. Only `san.se` is supported for geelm objects but `jack`, `j1s` or `fij` may be used for geeglm objects (if they have been estimated when fitting the model).
`...`	additional argument(s) for methods.

Value

A matrix (or vector) with columns giving lower and upper confidence limits for each parameter.

Functions

- `confint(geeglm)`:

drop1.geeglm — *Drop All Possible Single Terms to a* geeglm *Model Using Wald or Score Test*

Description

Compute all the single terms in the scope argument that can dropped from the model, and compute a table of the corresponding Wald test statistics.

Usage

```
## S3 method for class 'geeglm'
drop1(
  object,
  scope,
  test = c("Wald", "none", "score", "sasscore"),
  method = c("robust", "naive", "sandwich"),
  ...
)
```

Arguments

`object`	a fitted object of class geese.
`scope`	a formula giving the terms to be considered for adding or dropping.
`test`	the type of test to include.
`method`	Indicates which method is used for computing the standard error. `robust` is the default and corresponds to the modified sandwich estimator. `naive` is the classical naive variance estimate. `sandwich` is an alias for `robust`.
`...`	other arguments. Not currently used

Value

An object of class `anova` summarizing the differences in fit between the models.

Author(s)

Claus Ekstrom <`claus@ekstroem.dk`>

See Also

`drop1`, `geeglm`, `geese`

Examples

```
library(geepack)
data(ohio)
fit <- geeglm(resp ~ age + smoke + age:smoke, id=id, data=ohio,
            family=binomial, corstr="exch", scale.fix=TRUE)
drop1(fit)
```

---

drop1.geem *Drop All Possible Single Terms to a* geem *Model Using Wald or Score Test*

Description

Compute all the single terms in the scope argument that can dropped from the model, and compute a table of the corresponding Wald test statistics.

Usage

```
## S3 method for class 'geem'
drop1(
  object,
  scope,
  test = c("Wald", "none", "score", "sasscore"),
  method = c("robust", "naive", "sandwich"),
  ...
)
```

Arguments

object	a fitted object of class geese.
scope	a formula giving the terms to be considered for adding or dropping.
test	the type of test to include.
method	Indicates which method is used for computing the standard error. robust is the default and corresponds to the modified sandwich estimator. naive is the classical naive variance estimate. sandwich is an alias for robust.
...	other arguments. Not currently used

Value

An object of class anova summarizing the differences in fit between the models.

Author(s)

Claus Ekstrom <claus@ekstroem.dk>

See Also

drop1, geem

Examples

```
library(geeM)
library(geepack)
data(ohio)
fit <- geem(resp ~ age + smoke + age:smoke, id=id, data=ohio,
            family="binomial", corstr="exch", scale.fix=TRUE)
drop1(fit)
```

---

geelm.control *Control estimation of GEE models*

Description

Settings for controlling technical details of GEE fitting via geelm.

Usage

```
geelm.control(
  init.beta = NULL,
  init.phi = 1,
  tol = 1e-05,
  maxit = 20,
  scale.fix = FALSE,
  useP = TRUE,
  std.err = "san.se"
)
```

Arguments

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.phi	an optional initial overdispersion parameter. If not supplied, initialized to 1.
tol	Tolerance for asserting convergence.
maxit	integer giving the maximal number of Fisher Scoring iteration.
scale.fix	logical indicating if the scale should be fixed.
useP	If set to FALSE, do not use the n-p correction for dispersion and correlation estimates. This can be useful when the number of observations is small, as subtracting p may yield correlations greater than 1.
std.err	Character string specifying which standard error estimation method should be used. Supported options are san.se (sandwich SE) and naive.

Value

A list of values used for controlling model fitting.

`geelm.fit` *Fit Generalized Estimating Equation-based Linear Models*

Description

Estimate mean structure parameters and their corresponding standard errors for generalized linear models with clustered or correlated observations by use of generalized estimating equations.

Usage

```
geelm.fit(x, y, id, offset, family, weights, control, corstr, start = NULL)

geelm(
  formula,
  id = NULL,
  waves = NULL,
  data = parent.frame(),
  family = gaussian,
  corstr = "independence",
  Mv = 1,
  weights = NULL,
  corr.mat = NULL,
  offset = NULL,
  engine = "geeasy",
  output = "geelm",
  control = geelm.control()
)
```

Arguments

`x, y`	For `glm`: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For `glm.fit`: `x` is a design matrix of dimension `n * p`, and `y` is a vector of observations of length `n`.
`id`	A vector identifying the clusters. If NULL, then each observation is assigned its own cluster.
`offset`	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases. One or more `offset` terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See `model.offset`.

`family`	A description of the error distribution and link function to be used in the model. 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 Details below.
`weights`	an optional vector of 'prior weights' to be used in the fitting process. Should be `NULL` or a numeric vector.
`control`	A list of parameters for controlling the fitting process.
`corstr`	A character string specifying the correlation structure. The default is "independence". 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.
`start`	starting values for the parameters in the linear predictor.
`formula`	A formula expression similar to that for `glm`,
`waves`	An numeric vector identifying the time ordering within clusters (i.e. levels of `id`). 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. Note that only the ordering of the values in `waves` is used, NOT the numeric values themselves. This means that e.g. having waves equal to `c(1, 2, 3)` or `c(1, 2, 7)` within a cluster results in the same model.
`data`	An optional data frame containing the variables in the model.
`Mv`	For `"m-dependent"`, the value for `m`.
`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.
`engine`	Engine used to fit the model. The default, `"geeasy"` uses this package (built on the `geeM` package), while `"geepack"` uses the function `geeglm` from `geepack` to fit the model. Note that if the geepack engine is used, the data are sorted according to id (and possibly waves within id) and NAs are dropped before the data is used (this differs from the standard in geepack).
`output`	Output object type. There are two options; 1) `"geelm"` (default), resulting in an output that inherits the structure of geepacks `geeglm` object, or 2) `"geem"` (or its alias `"geeM"`) which results in an output that has the structure of `geeMs` `geem` object.

Details

Users may specify functions for link and variance functions, but the functions must be vectorized functions.

Offsets can be specified in the model formula, as in `glm()` or they may be specified using the `offset` argument. If offsets are specified in both ways, their sum is used as an offset.

For the `"userdefined"` correlation option, the function accepts a matrix with consecutive integers. Each such integer represent a distinct parameter that will be estimated. 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.`geelm` only looks at the upper triangle of the matrix. Any entry given as 0 will be fixed at 0.

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.

Concerning the `family` argument: 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 `geelm` 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.

Value

An object of class `geelm` (inherits from `geeglm`) representing the fit. It contains the following slots:

`$coefficients`: Coefficients from the mean structure model (betas) on their original scales

`$residuals`: Pearson residuals, in the order of the inputted dataset (with NAs omitted).

`$fitted.values`: Fitted values (response scale), in the order of the inputted dataset (with NAs omitted).

`$rank`: The rank of the model matrix, i.e. the number of estimated mean structure coefficients.

`$qr`: QR decomposition of the model matrix (NA omitted).

`$family`: A family object specifying which exponential family was used for fitting the mean structure model, see `family` for more information.

`$linear.predictors`: The linear predictor on the original scale.

`$weights`: Weights used for computations, in the order of the inputted dataset (NAs omitted).

`$prior.weights`: The original weights used to produce this geeglm object (set by user or defaulted to 1 for all observations).

`$df.residuals`: Residual degrees of freedom.

`$y`: Outcome variable, in the order of the inputted dataset (NAs omitted).

`$model`: The model.frame, ordered as the original inputted data with NAs omitted.

`$call`: The original function call that produced this geeglm object.

`$formula`: The formula used in the original call.

`$terms`: The terms of the formula used in the original call.

`$data`: The original dataset that was used for producing this geeglm object.

`$offset`: Offset used for fitting the model, ordered as the original inputted data with NAs omitted.

`$control`: Value of control parameters used for fitting the model.

`$method`: Internal function used for fitting the model.

`$contrasts`: Contrasts used in the model matrix.

`$xlevels`: Levels of factor variables used in the model formula (if any).

`$geese`: An object containing further information about the variance estimation, including a variance matrix for the beta-coefficients (`$vbeta`), the estimated coefficients for the working correlation matrix (`$alpha`), the estimated dispersion parameter (`$gamma`), and the individual cluster sizes (`$clusz`). See `geese` for more information.

`$modelInfo`: Information about the link functions used for fitting the mean, variance and scale structures of the model.

`$id`: IDs used for identifying the clusters, ordered as the original inputted data with NAs omitted.

`$corstr`: Name of the correlation structured imposed on the model. If the correlation structure requires further information, it is stored in a suitably named attribute. For example, for m-dependent correlation structures, the m scalar is available in an attribute named `Mv`.

`$cor.link`: Link function used for the correlation structure.

`$std.err`: Method used to estimate the standard error of the mean structure coefficients (betas).

Functions

- `geelm.fit()`:

Author(s)

Anne Helby Petersen, Lee McDaniel & Nick Henderson

See Also

`glm`, `formula`, `family`

Examples

```
# load data
data("respiratory")
respiratory$useid <- interaction(respiratory$center, respiratory$id)

# fit model
m <- geelm(outcome ~ treat + sex + age + baseline,
           data = respiratory, id = useid,
                      family = "binomial", corstr = "exchangeable")

## Not run:
get_jack_se <- function(object, dat){
    parm <- sapply(1:nrow(dat),
                   function(i){
                       dat.i <- dat[-i,]
                       coef(update(object, data=dat.i))
                   })
    parm <- t(parm)
    parm.mean <- apply(parm, 2, mean)

    parm.cent <- sapply(1:nrow(parm),
                        function(i){
                            parm[i, ] - parm.mean
                        })
    parm.cent <- t(parm.cent)

    jack.var <- ((nrow(dat)-1) / nrow(dat)) * t(parm.cent) %*% parm.cent
    jack.se <- sqrt(diag(jack.var))
    jack.se
}


# load data
data("respiratory")
respiratory$useid <- interaction(respiratory$center, respiratory$id)

# fit model
obj <- geelm(outcome ~ treat + sex + age + baseline,
           data = respiratory, id = useid,
                      family = "binomial", corstr = "exchangeable")

dat <- respiratory
get_jack_se(obj, dat)
summary(obj) |> coef()

## End(Not run)
```

getGEE *Get information for a geelm/geeglm object*

Description

Get information for a geelm/geeglm object

Usage

```
getGEE(object, name, ...)

getME(object, name, ...)
```

Arguments

`object`	A geeglm model object as obtained from `geepack::geeglm()` or a geelm object as obtained from `geeasy::geelm()`
`name`	Name of the slot/component of the geelm/geeglm object that should be returned. See list of possible names in details below.
`...`	Any additional arguments passed on to other functions.

Details

The allowed names are:

`coefficients` or `beta`: Coefficients from the mean structure model (betas) on their original scales

`residuals`: Pearson residuals, in the order of the inputted dataset (with NAs omitted).

`fitted.values`: Fitted values (response scale), in the order of the inputted dataset (with NAs omitted).

`rank`: The rank of the model matrix, i.e. the number of estimated mean structure coefficients.

`family`: A family object specifying which exponential family was used for fitting the mean structure model, see `family` for more information.

`linear.predictors`: The linear predictor on the original scale.

`df.residuals`: Residual degrees of freedom.

`corstr`: Name of the correlation structured imposed on the model. If the correlation structure requires further information, it is stored in a suitably named attribute. For example, for m-dependent correlation structures, the m scalar is available in an attribute named `Mv`.

`std.err`: Method used to estimate the standard error of the mean structure coefficients (betas).

`alpha`: The estimated parameter(s) for the variance structure.

`gamma` or `dispersion`: The estimated dispersion parameter.

`vbeta`: The estimated variance matrix of the mean structure (beta) coefficients.

`beta.se`: The standard errors of the mean structure (beta) coefficients.

`clusz` or `clustersizes`: Sizes of each of the clusters in the data.

`nclusters`: The total number of clusters.

Value

The requested slot of the object.

Functions

- `getME()`:

`plotEst` *Plot parameter estimates with 95 pct confidence intervals.*

Description

Parameter estimates are plotted along with error bars indicating 95 pct confidence intervals.

Usage

```
plotEst(..., intercept = TRUE, par = NULL, colors = NULL)
```

Arguments

`...`	One or more models. Currently, `lm`, `glm`, `geeglm`, `geelm`, and `mice` models are supported, but other model types may be applicable as well, see details.
`intercept`	Logical indicating whether the intercept should be plotted (defaults to `TRUE`).
`par`	Which model parameters to plot estimates for, given as character strings. Default is `NULL` which means that all parameter estimates are plotted.
`colors`	Color scale to use if several models are plotted. Defaults to a color blind friendly scale. If there are more models than there are colors, the color values are repeated.

Details

One or more models can be supplied, and if the parameters have the same names across models, they will be grouped together allowing for easy comparison.

Note that models can be given with or without names. If names are supplied (see example below), these are printed in the plot legend. Otherwise, the name of the model object is printed there instead.

Implementation details: As a default, the estimates are extracted from the `$coefficients` slot from the model object and confidence intervals are computed by calling `confint()`. This means that `plotEst` supports all models that have a `coefficents` slot and a `confint` method.

Value

No return values; called for side effects.

Author(s)

Anne Helby Petersen

Examples

```
# Fit example models
data(iris)
m1 <- lm(Sepal.Length ~ Petal.Length, iris)
m2 <- lm(Sepal.Length ~ Petal.Length + Petal.Width, iris)

# Plot one model
plotEst(m2)

# Plot two models with default model labels
plotEst(m1, m2)

# Plot two models with custom model labels (simple)
plotEst(model1 = m1, model2 = m2)

# Plot two models with custom model labels (with spacing)
plotEst(`Simple model` = m1, `Full petal model` = m2)

# Plot two models without intercept
plotEst(m1, m2, intercept = FALSE)
```

```
# Plot two models with custom parameter subset
plotEst(m1, m2, par = c("Petal.Length"))

# Plot two models with custom color scale given by color names
plotEst(m1, m2, colors = c("red", "blue"))

# Plot two models with custom color scale given by color hex codes
#  note: only the first colors are used as there are more
#  colors than models
plotEst(m1, m2, colors = c("#CC6666", "#9999CC", "#66CC99"))
```

`scorefct` *Internal functions for the MESS package*

Description

Internal functions for the MESS package

Usage

```
scorefct(object, beta = NULL, testidx = NULL, sas = FALSE)
```

Arguments

`object`	input geepack object from a geeglm fit.
`beta`	The estimated parameters. If set to `NULL` then the parameter estimates are extracted from the model fit object object.
`testidx`	Indices of the beta parameters that should be tested equal to zero
`sas`	Logical. Should the SAS version of the score test be computed. Defaults to `FALSE`.

Author(s)

Claus Ekstrom `<claus@ekstroem.dk>`

Index