Package 'robustvarComp'

April 18, 2019

Title Robust Estimation of Variance Component Models

LazyLoad yes

LazyData yes

Version 0.1-5

Author Claudio Agostinelli <claudio.agostinelli@unitn.it> and Victor J. Yohai <victoryohai@gmail.com>

Maintainer Claudio Agostinelli <claudio.agostinelli@unitn.it>

Depends R (>= 2.15.1)

Imports robustbase, GSE, numDeriv, robust, plyr

Suggests nlme, Matrix, mvtnorm, WWGbook

Date 2019-04-01

Description Robust Estimation of Variance Component Models by classic and composite robust procedures. The composite procedures are robust against outliers generated by the Independent Contamination Model.

License GPL-2

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-04-18 06:10:03 UTC

R topics documented:

summary.varComprob *Summary Method for "varComprob" Objects*

Description

Summary method for R object of class "varComprob".

Usage

```
  ## S3 method for class 'varComprob'
summary(object, print.outliers = FALSE, ...)
```

Arguments

object	an R object of class varComprob, typically created by varComprob.
print.outliers	logical. If TRUE three tables are also printed, summarizing the presence of cell, couple and row outliers in the data set.
...	potentially more arguments passed to methods.

Value

summary(object) returns an object of S3 class "summary.varComprob", basically a list.

Author(s)

Claudio Agostinelli and Victor J. Yohai

Examples

```
if (!require(nlme))
  stop()
data(Orthodont)

z1 <- rep(1, 4)
z2 <- c(8,10,12,14)
K <- list()
K[[1]] <- tcrossprod(z1,z1) ## Int
K[[2]] <- tcrossprod(z1,z2) + tcrossprod(z2,z1) ## Int:age
K[[3]] <- tcrossprod(z2,z2) ## age
names(K) <- c("Int", "Int:age", "age")
p <- 4
n <- 27
groups <- cbind(rep(1:p, each=n), rep((1:n), p))

## Composite Tau
OrthodontCompositeTau <- varComprob(distance ~ age*Sex, groups = groups,
  data = Orthodont, varcov = K,
  control=varComprob.control(lower=c(0,-Inf,0)))

summary(OrthodontCompositeTau, print.outliers=TRUE)
```

varComprob *Fitting variance component models using robust procedures*

Description

varComprob and varComprob.fit fit linear mixed-effect models where the marginal variance-covariance matrix is linear in known positive semidefinite matrices. varComprob uses a formula interface, whereas varComprob.fit is the underlying working horse.

Usage

```
varComprob(fixed, data, random, groups, varcov, weights, subset,
  family = stats::gaussian("identity"), na.action, offset,
  control = varComprob.control(...), doFit = TRUE,
  normalizeTrace = FALSE, contrasts = NULL,
  model = TRUE, X = TRUE, Y = TRUE, K = TRUE, ...)

varComprob.fit(Y, X, V, control = varComprob.control(), ...)
```

Arguments

fixed	A two-sided formula specifying the fixed effects of the model. This is the same as in stats::lm.
data	An optional data frame, list or environment containing the variables in the model. If not found in data, the variables are taken from environment(fixed), typically the environment from which varComprob is called. This is the same as the data argument of stats::lm.
random	This argument is not yet used.
groups	a numeric matrix with two columns which is used to group appropriately the observations according to subject. See details below and the example.
varcov	An list of symmetric positive semidefinite matrices. The weighted sum of these matrices represent the contribution of random effects to the marginal variance of the response variable, with unknown weights representing variance components.
weights	This argument is not yet used.
subset	An optional vector specifying a subset of observations to be used in the fitting process.
family	The same as the family argument of stats::glm. However, only gaussian('identity') is supported currently.
na.action	The same as in stats::lm.
offset	The same as in stats::glm. These offsets are assumed as known fixed effects.
control	Usually an object from calling varComprob.control.
doFit	This argument is not yet used.
normalizeTrace	A logical scalar, indicating whether the individual variance-covariance matrices should be normalized such that variance components are on the same scale.

contrasts	The same as in stats::lm.
model	A logical scalar, indicating whether the model frame will be included in the result.
X	For varComprob, this is a logical scalar, indicating whether the fixed-effect design matrix should be included in the result. For varComprob.fit this is the optional numeric array containing the fixed effect design matrix for the model, see details below. If X is missing or an array with zero dimension, it is assumed that Y has zero mean.
Y	For varComprob, this is a logical scalar, indicating whether the response variable should be included in the result. For varComprob.fit, this is a numeric matrix of response variables. See details below.
K	This is a logical scalar, indicating whether the list of variance-covariance matrices should be included in the result. These matrices are those specified by varcov. See details below.
V	This is a numeric array containing the variance-covariance matrices. See details below.
...	When control is given, this is ignored. Otherwise, these arguments are passed to varComprob.control and the result will be used as the control argument.

Details

The variance component model is of form

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}$$

where $\mathbf{e}$ is multivariate normally distributed with mean zero and variance-covariance matrix $\boldsymbol{\Sigma}$ being

$$\boldsymbol{\Sigma} = \sum_{j=1}^{R} \sigma_j^2 \mathbf{K}_j + \sigma_e^2 \mathbf{I}$$

in which $\mathbf{K}_j$ are known positive semidefinite matrices and $\mathbf{I}$ is the identity matrix.

In the varComprob formula interface, the $\mathbf{X}$ matrix and response variable are specified by the fixed argument. The varcov argument specifies each variance-covariance matrix in a list.

In varComprob.fit, let p the number of observations, n the number of independent replicates and k the number of regressors. Then Y must be a matrix of dimension $p \times n$, X must be an array of dimension $p \times n \times k$ and V must be an array of dimension $p \times p \times R$, where R is the number of variance-covariance matrices.

The model fitting process is performed by a robust procedures. See references for more details.

See varComprob.control for arguments controlling the modeling fitting.

Value

The value of any of the two functions is a list with class varComprob or varComprob.fit respectively. The following elements are present in both

- call: the actual call.

- beta: a named numeric vector of parameter estimates. These are the estimated of the fixed effect parameters.
- vcov.beta: estimated variance-covariance matrix of the estimated fixed effects parameters.
- eta: a named numeric vector of parameter estimates. These are the estimated variance components.
- vcov.eta: estimated variance-covariance matrix of the estimated random effects variance parameters.
- gamma: a named numeric vector of parameter estimates. These are the estimated ratio of each variance component relative to the error variance.
- vcov.gamma: estimated variance-covariance matrix of the estimated ratio of random effects variance parameters with respect the estimated error variance.
- eta0: the estimated error variance.
- resid: residuals in matrix form $p \times n$.
- weights: final weights of the iterative fixed point equations.
- dotweights: another type of weight. See references for details.
- Sigma: an estimates of the variance-covariance marginal matrix.
- scales: the scales in case of composite robust procedures otherwise NULL.
- scale: the scale in case of classical robust procedures otherwise NULL.
- min: the minimum attains by the goal function.
- scale0:
- initial.values: a list with the following components: beta: initial value for the fixed parameters; gamma: initial value for the ratio of each variance component relative to the error variance; eta0: initial value for the errot variance; scales: initial value for the scales in case of composite robust methods otherwise is not available; scale: initial value for the scale in case of classic robust methods otherwise is not available.
- iterations: number of iterations.
- control: the control argument.
- method: the robust method used to perform the estimation.

The function varComprob returns also

- fixed: the same as beta.
- parms: the same as gamma.
- sigma2: the same as eta0.
- nobs: the number of observations.
- na.action: the na.action used in the model frame.
- offset: the same as input.
- contrasts: the contrast used in the fixed-effect design matrix.
- random.labels: the labels used to differentiate random effects. Note that the error variance is not included here. It is safe to check the number of variance components specified by the model by checking the length of random.labels.

Author(s)

Claudio Agostinelli and Victor J. Yohai

References

C. Agostinelli and V.J. Yohai (2014) Composite Robust Estimators for Linear Mixed Models. arXiv:1407.2176.

M.P. Victoria-Feser and Copt (2006) High Breakdown Inference in the Mixed Linear Model. Journal of American Statistical Association, 101, 292-300.

Examples

```
  if (!require(nlme))
    stop()
  data(Orthodont)

  z1 <- rep(1, 4)
  z2 <- c(8,10,12,14)
  K <- list()
  K[[1]] <- tcrossprod(z1,z1) ## Int
  K[[2]] <- tcrossprod(z1,z2) + tcrossprod(z2,z1) ## Int:age
  K[[3]] <- tcrossprod(z2,z2) ## age
  names(K) <- c("Int", "Int:age", "age")
  p <- 4
  n <- 27
  groups <- cbind(rep(1:p, each=n), rep((1:n), p))

  ## Not run:

  ## Composite S
  OrthodontCompositeS <- varComprob(distance ~ age*Sex, groups = groups,
      data = Orthodont, varcov = K,
      control=varComprob.control(method="compositeS", lower=c(0,-Inf,0)))

## End(Not run)

  ## Composite Tau
  OrthodontCompositeTau <- varComprob(distance ~ age*Sex, groups = groups,
      data = Orthodont, varcov = K,
      control=varComprob.control(lower=c(0,-Inf,0)))

  ## Not run:

  summary(OrthodontCompositeTau)

  ## Classic S
  OrthodontS <- varComprob(distance ~ age*Sex, groups = groups,
      data = Orthodont, varcov = K,
      control=varComprob.control(lower=c(0,-Inf,0),
      method="S", psi="rocke"))
```

```
  summary(OrthodontS)

## End(Not run)
  ## Not run:

  if (!require(WWGbook))
    stop()
  if (!require(nlme))
    stop()
  data(autism)
  autism <- autism[complete.cases(autism),]
  completi <- table(autism$childid)==5
  completi <- names(completi[completi])
  indici <- as.vector(unlist(sapply(completi,
              function(x) which(autism$childid==x))))
  ind <- rep(FALSE, nrow(autism))
  ind[indici] <- TRUE
  autism <- subset(autism, subset=ind) ## complete cases 41
  attach(autism)
  sicdegp.f <- factor(sicdegp)
  age.f <- factor(age)
  age.2 <- age - 2
  sicdegp2 <- sicdegp
  sicdegp2[sicdegp == 3] <- 0
  sicdegp2[sicdegp == 2] <- 2
  sicdegp2[sicdegp == 1] <- 1
  sicdegp2.f <- factor(sicdegp2)
  autism.updated <- subset(data.frame(autism,
                    sicdegp2.f, age.2), !is.na(vsae))
  autism.grouped <- groupedData(vsae ~ age.2 | childid,
                    data=autism.updated, order.groups = FALSE)
  p <- 5
  n <- 41
  z1 <- rep(1, p)
  z2 <- c(0, 1, 3, 7, 11)
  z3 <- z2^2
  K <- list()
  K[[1]] <- tcrossprod(z1,z1)
  K[[2]] <- tcrossprod(z2,z2)
  K[[3]] <- tcrossprod(z3,z3)
  K[[4]] <- tcrossprod(z1,z2) + tcrossprod(z2,z1)
  K[[5]] <- tcrossprod(z1,z3) + tcrossprod(z3,z1)
  K[[6]] <- tcrossprod(z3,z2) + tcrossprod(z2,z3)
  names(K) <- c("Int", "age", "age2", "Int:age", "Int:age2", "age:age2")

  groups <- cbind(rep(1:p, each=n), rep((1:n), p))

  ## Composite Tau
  AutismCompositeTau <- varComprob(vsae ~ age.2 + I(age.2^2)
    + sicdegp2.f + age.2:sicdegp2.f + I(age.2^2):sicdegp2.f,
    groups = groups,
    data = autism.grouped, varcov = K,
    control=varComprob.control(
```

```
  lower=c(0.01,0.01,0.01,-Inf,-Inf,-Inf)))

summary(AutismCompositeTau)

## Classic S
AutismS <- varComprob(vsae ~ age.2 + I(age.2^2)
  + sicdegp2.f + age.2:sicdegp2.f + I(age.2^2):sicdegp2.f,
  groups = groups,
  data = autism.grouped, varcov = K,
  control=varComprob.control(
  method="S", psi="rocke", cov.init="covOGK",
  lower=c(0.01,0.01,0.01,-Inf,-Inf,-Inf)))
summary(AutismS)

## End(Not run)
```

`varComprob.control` *Tuning Parameters for* `varComprob()` *and Auxiliaries*

Description

Tuning Parameters for `varComprob()` which performs S, composite S and Tau estimators for variance component models.

Usage

```
varComprob.control(init = NULL, lower = 0, upper = Inf, epsilon = 0.001,
  tuning.chi = NULL, bb = 0.5, tuning.psi = NULL,
  arp.chi = 0.1, arp.psi = NULL, max.it = 100,
  rel.tol.beta = 1e-06, rel.tol.gamma = 1e-05, rel.tol.scale = 1e-05,
  trace.lev = 0,
  method = c("compositeTau", "compositeS", "compositeMM",
    "Tau", "S", "MM"),
  psi = c("optimal", "bisquare", "rocke"),
  beta.univ = FALSE, gamma.univ = FALSE,
  fixed.init = c("lmrob.S", "lmRob"),
  cov.init = c("TSGS", "2SGS", "covOGK"),
  cov = TRUE, ...)
```

Arguments

`init`	A list with initial values. The only components used are `beta`, `gamma`, `eta0`, `scales` (in case of composite methods), `scale` (in case of classic methods) and `Sigma`. If one of the components is empty a suitable automatic initial values is calculated.
`lower`	A numeric vector with length equals to `gamma`. This parameter is passed to the `optim` function.

upper	A numeric vector with length equals to gamma. This parameter is passed to the optim function.
epsilon	A positive numeric scalar. This value is set to the non negative elements of gamma (checked by lower) when the automatic initial values are negative.
tuning.chi	tuning constant vector for the rho_1 function. If NULL, as by default, this is set, depending on "psi" (for now only "bisquare" and "optimal"), to a suitable value.
bb	expected value under the normal model of the rho function with tuning constant equal to tuning.chi.
tuning.psi	tuning constant vector for the rho_2 function. If NULL, as by default, this is set, depending on "psi" (for now only "bisquare" and "optimal"), to a suitable value.
arp.chi	tuning constant vector for the rho_1 function in case psi is set to "rocke".
arp.psi	tuning constant vector for the rho_2 function in case psi is set to "rocke".
max.it	integer specifying the maximum number of IRWLS iterations.
rel.tol.beta	(for the RWLS iterations algorithm of the fixed parameters): relative convergence tolerance for the parameter vector.
rel.tol.gamma	(for the optim function used in the estimation procedure of the random variance parameters): relative convergence tolerance for the parameter vector.
rel.tol.scale	relative convergence tolerance for the scale vector.
trace.lev	integer indicating if the progress of the algorithm should be traced (increasingly); default 'trace.lev = 0' does no tracing.
method	string specifying the estimator-chain. For now available procedures are 'compositeS', 'compositeTau' and 'S'. Default is set to 'compositeTau'.
psi	string specifying the type psi-function used. Available choices for the composite methods are "bisquare" and "optimal". For classic methods "rocke", "bisquare" and "optimal". Defaut is set to "optimal".
beta.univ	logical. If TRUE a robust simple regression is performed for each explanatory variable in order to get starting values for the fixed effect parameters.
gamma.univ	logical. If TRUE a simple regression is performed for each explanatory variable in order to get starting values for the random variance parameters.
fixed.init	string with function name to be used to calculate initial value of the fixed effect parameters. Possible values are "lmrob.S" and "lmRob".
cov.init	function or string with function name to be used to calculate initial covariance matrix estimate if necessary. Possible string value is "TSGS", "2SGS" and "covOGK". Default is set to "TSGS".
cov	logical. If TRUE the estimated variance-covariance matrix for the fixed and random parameters is reported.
...	further arguments.

Value

returns a named 'list' with over twenty components, corresponding to the arguments.

Author(s)

Claudio Agostinelli and Victor J. Yohai

Examples

```
## Show the default settings:
  str(varComprob.control())
```

Index