Package 'mecor'

January 14, 2021

Type Package

Title Measurement Error Correction in Linear Models with a Continuous Outcome

Version 0.9.0

Author Linda Nab

Maintainer Linda Nab <lindanab4@gmail.com>

Description Covariate measurement error correction is implemented by means of regression calibration by Carroll RJ, Ruppert D, Stefanski LA & Crainiceanu CM (2006, ISBN:1584886331), efficient regression calibration by Spiegelman D, Carroll RJ & Kipnis V (2001) <doi:10.1002/1097-0258(20010115)20:1%3C139::AID-SIM644%3E3.0.CO;2-K> and maximum likelihood estimation by Bartlett JW, Stavola DBL & Frost C (2009) <doi:10.1002/sim.3713>. Outcome measurement error correction is implemented by means of the method of moments by Buonaccorsi JP (2010, ISBN:1420066560) and efficient method of moments by Keogh RH, Carroll RJ, Tooze JA, Kirkpatrick SI & Freedman LS (2014) <doi:10.1002/sim.7011>. Standard error estimation of the corrected estimators is implemented by means of the Delta method by Rosner B, Spiegelman D & Willett WC (1990) <doi:10.1093/oxfordjournals.aje.a115715> and Rosner B, Spiegelman D & Willett WC (1992) <doi:10.1093/oxfordjournals.aje.a116453>, the Fieller method described by Buonaccorsi JP (2010, ISBN:1420066560), and the Bootstrap by Carroll RJ, Ruppert D, Stefanski LA & Crainiceanu CM (2006, ISBN:1584886331).

Depends R (>= 2.10)

License GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/LindaNab/mecor

Imports lme4, lmerTest, numDeriv

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-01-14 09:20:02 UTC

R topics documented:

`ccs` *Covariate-Calibration Study*

Description

A simulated dataset containing 1000 observations of the outcome Y, the error-prone exposure X_star with systematic measurement error, the covariate Z and two replicates measures X1_star and X2_star of the exposure with classical measurement error. The two replicates are observed in the first 500 study participants.

Usage

```
ccs
```

Format

A data frame with 1000 rows and 5 variables:

Y outcome, continuous

X_star error-prone exposure with systematic measurement error, continuous

Z covariate, continuous

X_star_1 first replicate of error-prone exposure with classical measurement error, continuous

X_star_2 second replicate of error-prone exposure with classical measurement error, continuous

Examples

```
data("ccs", package = "mecor")
```

ecvs *External Covariate-Validation Study*

Description

A simulated dataset containing 100 observations of the gold standard measurement of the exposure X, the error-prone exposure X_star and the covariate Z. The outcome Y is not observed in the study population. To be used in combination with the dataset icvs.

Usage

```
ecvs
```

Format

A data frame with 100 rows and 3 variables:

X gold standard exposure, continuous

X_star error-prone exposure, continuous

Z covariate, continuous

Examples

```
data("ecvs", package = "mecor")
```

eovs *External Outcome-Validation Study*

Description

A simulated dataset containing 100 observations of the gold standard measurement of outcome Y and the error-prone outcome Y_star. The covariates X and Z are not observed in the study population. To be used in combination with the dataset iovs.

Usage

```
eovs
```

Format

A data frame with 100 rows and 2 variables:

Y gold standard outcome, continuous

Y_star error-prone outcome, continuous

Examples

```
data("eovs", package = "mecor")
```

icvs *Internal Covariate-Validation Study*

Description

A simulated dataset containing 1000 observations of the outcome Y, the error-prone exposure X_star and the covariate Z. The gold standard exposure X is observed in approximately 25% of the study population

Usage

```
icvs
```

Format

A data frame with 1000 rows and 4 variables:

Y outcome, continuous

X_star error-prone exposure, continuous

Z covariate, continuous

X gold standard exposure, continuous

Examples

```
data("icvs", package = "mecor")
```

iovs *Internal Outcome-Validation Study*

Description

A simulated dataset containing 1000 observations of the error-prone outcome Y_star, the exposure X and the covariate Z. The gold standard outcome Y is observed in approximately 25% of the study population

Usage

```
iovs
```

Format

A data frame with 1000 rows and 4 variables:

Y_star error-prone outcome, continuous

X exposure, continuous

Z covariate, continuous

Y gold standard outcome, continuous

#' @examples data("iovs", package = "mecor")

`iovs_diff` *Internal Outcome Validation Study*

Description

A simulated dataset containing 1000 observations of the error-prone outcome Y_star with differential measurement error, and the binary exposure X. The gold standard outcome Y is observed in approximately 25% of the study population

Usage

```
iovs_diff
```

Format

A data frame with 1000 rows and 3 variables:

Y_star error-prone outcome, continuous

X exposure, binary

Y gold standard outcome, continuous

Examples

```
data("iovs_diff", package = "mecor")
```

ipwm *Weighting for Confounding and Joint Misclassification of Exposure and Outcome*

Description

ipwm implements a method for estimating the marginal causal odds ratio by constructing weights (modified inverse probability weights) that address both confounding and joint misclassification of exposure and outcome.

Usage

```
ipwm(
  formulas,
  data,
  outcome_true,
  outcome_mis = NULL,
  exposure_true,
  exposure_mis = NULL,
  nboot = 1000,
  conf_level = 0.95,
  fix_nNAs = FALSE,
  semiparametric = FALSE,
  optim_args = list(method = "BFGS"),
  force_optim = FALSE,
  sp = Inf,
  print = TRUE
)
```

Arguments

formulas	a list of objects of class formula specifying the probability models for the stats::terms of some factorisation of the joint conditional probability function of exposure_true, exposure_mis, outcome_true and outcome_mis, given covariates
data	data.frame containing exposure.true, exposure.mis, outcome.true, outcome.mis and covariates. Missings (NAs) are allowed on variables exposure_true and outcome_true.
outcome_true	a character string specifying the name of the true outcome variable that is free of misclassification but possibly unknown (NA) for some (but not all) subjects
outcome_mis	a character string specifying the name of the counterpart of outcome_true that is available on all subjects but potentially misclassifies subjects' outcomes. The default (outcome_mis = NULL) indicates absence of outcome misclassification
exposure_true	a character string specifying the name of the true exposure variable that is free of misclassification but possibly unknown (NA) for some (but not all) subjects

exposure_mis	a character string specifying the name of the counterpart of exposure_true that is available on all subjects but potentially misclassifies subjects as exposed or as non-exposed. The default (exposure_mis = NULL) indicates absence of exposure misclassification
nboot	number of bootstrap samples. Setting nboot == 0 results in point estimation only.
conf_level	the desired confidence level of the confidence interval
fix_nNAs	logical indicator specifying whether or not to fix the joint distribution of is.na(exposure_true) and is.na(outcome_true). If TRUE, stratified bootstrap sampling is done according to the missing data pattern.
semiparametric	logical indicator specifying whether or not to parametrically sample exposure_true, exposure_mis, outcome_true and outcome_mis. If semiparametric == TRUE, it is assumed that the missing data pattern is conditionally independent of these variables given covariates. Provided nboot > 0, the missing data pattern and covariates are sampled nonparametrically. semiparametric is ignored if nboot == 0.
optim_args	arguments passed onto optim if called. See Details below for more information.
force_optim	logical indicator specifying whether or not to force the optim function to be called
sp	scalar shrinkage parameter in the interval (0,Inf). Values closer to zero result in greater shrinkage of the estimated odds ratio to unity; sp == Inf results in no shrinkage.
print	logical indicator specifying whether or not to print the output.

Details

This function is an implementation of the weighting method described by Penning de Vries et al. (2018). The method defaults to the estimator proposed by Gravel and Platt (2018) in the absence of exposure misclassification.

The function assumes that the exposure or the outcome has a misclassified version. An error is issued when both outcome_mis and exposure_mis are set to NULL.

Provided force_optim = FALSE, ipwm is considerably more efficient when the optim function is not invoked; i.e., when (1) exposure_mis = NULL and the formula for outcome_true does not contain stats::terms involving outcome_mis or exposure_true, (2) outcome_mis = NULL and the formula for exposure_true does not contain stats::terms involving exposure_mis or outcome_true, or (3) all(is.na(data[,exposure_true]) == is.na(data[,outcome_true])) and the formulas for exposure_true and outcome_true do not contain stats::terms involving exposure_mis or outcome_mis. In these cases, ipwm uses iteratively reweighted least squares via the glm function for maximum likelihood estimation. In all other cases, optim_args is passed on to optim for optimisation of the joint likelihood of outcome_true, outcome_mis, exposure_true and exposure_mis.

Value

ipwm returns an object of class ipwm. The returned object is a list containing the following elements:

logOR	the estimated log odds ratio;

`call` the matched function call.

If `nboot != 0`, the list also contains

`SE` a bootstrap estimate of the standard error for the estimator of the log odds ratio;

`CI` a bootstrap percentile confidence interval for the log odds ratio.

Author(s)

Bas B. L. Penning de Vries, <b.b.l.penning_de_vries@lumc.nl>

References

Gravel, C. A., & Platt, R. W. (2018). Weighted estimation for confounded binary outcomes subject to misclassification. *Statistics in medicine*, 37(3), 425-436. https://doi.org/10.1002/sim.7522

Penning de Vries, B. B. L., van Smeden, M., & Groenwold, R. H. H. (2020). A weighting method for simultaneous adjustment for confounding and joint exposure-outcome misclassifications. *Statistical Methods in Medical Research*, 0(0), 1-15. https://doi.org/10.1177/0962280220960172

Examples

```
data(sim) # simulated data on 10 covariates, exposure A and outcome Y.
formulas <- list(
  Y ~ A + L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + L9 + L10 + B + Z,
  A ~ L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + L9 + L10 + B + Z,
  Z ~ L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + L9 + L10 + B,
  B ~ L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + L9 + L10
)
## Not run:
ipwm_out <- ipwm(
  formulas = formulas,
  data = sim,
  outcome_true = "Y",
  outcome_mis = "Z",
  exposure_true = "A",
  exposure_mis = "B",
  nboot = 200,
  sp = 1e6
)
ipwm_out

## End(Not run)
```

MeasError *Create a Measurement Error Object*

Description

This function creates a measurement error object, usually used as a covariate or the outcome in the `formula` argument of mecor if one wants to correct for the measurement error in that variable using a reference variable or a replicate measure.

Usage

```
MeasError(substitute, reference, replicate, differential)
```

Arguments

`substitute`	a vector containing the error-prone measure
`reference`	a vector containing the reference measure assumed without measurement error
`replicate`	a vector or matrix with replicates of the error-prone measure with classical measurement error. This can either be replicates obtained by using the same measurement method as the substitute measure (replicates study) or replicates using a different measurement method than the substitute measure (calibration study).
`differential`	a vector containing the variable to which the measurement error is differential.

Value

`MeasError` returns an object of class "MeasError".

An object of class `MeasError` is a list containing the substitute and reference (and replicate or differential if applicable) variables and has attributes input (the name of the substitute and reference or replicate and differential (if applicable) variables) and call (the matched call).

Author(s)

Linda Nab, <l.nab@lumc.nl>

Examples

```
## measurement error in a covariate:
# internal covariate-validation study
data(icvs)
with (icvs, MeasError(substitute = X_star,
                      reference = X))
# replicates study
data(rs)
with (rs, MeasError(substitute = X_star_1,
                    replicate = cbind(X_star_2, X_star_3)))
# covariate-calibration study
data(ccs)
```

```
with(ccs, MeasError(substitute = X_star,
                    replicate = cbind(X_star_1, X_star_2)))
## measurement error in the outcome:
# internal outcome-validation study
data(iovs)
with(iovs, MeasError(substitute = Y_star,
                     reference = Y))
# internal outcome- validation study with differential measurement error in
# the dependent variable
data(iovs_diff)
with(iovs_diff, MeasError(substitute = Y_star,
                          reference = Y,
                          differential = X))
```

`MeasErrorExt` *Create an External Measurement Error Object*

Description

This function creates an external measurement error object, usually used as a covariate or the outcome in the `formula` argument of mecor if one wants to correct for the measurement error in that variable using external data or externally estimated coefficients of the calibration model (covariate-measurement error) or measurement error model (outcome-measurement error)

Usage

```
MeasErrorExt(substitute, model)
```

Arguments

`substitute`	a vector containing the error-prone measure
`model`	a fitted linear model of class lm or a named list. The list contains a vector named `coef`: the coefficients of the calibration model or measurement error model and an optional matrix named `vcov`: the variance–covariance matrix of the coefficients

Value

`MeasErrorExt` returns an object of class "MeasErrorExt".

An object of class `MeasErrorExt` is a list containing the substitute variable and the fitted calibration model or measurement error model and has attributes input (the name of the substitute variable) and call (the matched call).

Author(s)

Linda Nab, <l.nab@lumc.nl>

Examples

```
## measurement error in a covariate:
# external covariate-validation study
data(ecvs)
# calibration model
calmod_fit <- lm(X ~ X_star + Z, data = ecvs)
# the external covariate-validation study can be used to correct for the
# measurement error in X_star in the dataset 'icvs', using the fitted
# calibration model
data(icvs)
with (icvs, MeasErrorExt(substitute = X_star,
                         model = calmod_fit))
# identical to:
calmod_coef <- coefficients(calmod_fit)
calmod_vcov <- vcov(calmod_fit)
with (icvs, MeasErrorExt(substitute = X_star,
                         model = list(coef = calmod_coef,
                                      vcov = calmod_vcov)))
# when no external data is available, guesstimations of the coefficients of
# the calibration model can be used instead:
with (icvs, MeasErrorExt(substitute = X_star,
                         model = list(coef = c('(Intercept)' = 0,
                                               'X_star' = 0.8,
                                               'Z' = 0.2))))
## measurement error in the outcome:
# external outcome-validation study
data(eovs)
memod_fit <- lm(Y_star ~ Y, data = eovs)
# the external outcome-validation study can be used to correct for the
# measurement error in Y_star in the dataset 'iovs', using the fitted
# measurement error model
with (iovs, MeasErrorExt(substitute = Y_star,
                         model = memod_fit))
# identical to:
memod_coef <- coefficients(memod_fit)
memod_vcov <- vcov(memod_fit)
with (iovs, MeasErrorExt(substitute = Y_star,
                         model = list(coef = memod_coef,
                                      vcov = memod_vcov)))
# when no external data is available, guesstimations of the coefficients of
# the measurement error model can be used instead:
with (iovs, MeasErrorExt(substitute = Y_star,
                         model = list(coef = c('(Intercept)' = 0,
                                               'Y' = 0.5))))
```

`MeasErrorRandom` *Create a Random Measurement Error Object*

Description

This function creates a random measurement error object, usually used as a covariate in the `formula` argument of mecor if one wants to correct for random measurement error in that variable

Usage

```
MeasErrorRandom(substitute, variance)
```

Arguments

`substitute`	a vector containing the error-prone measure
`variance`	a numeric quantifying the assumed variance of the random measurement error

Value

`MeasErrorRandom` returns an object of class "MeasErrorRandom".

An object of class `MeasErrorRandom` is a list containing the substitute variable, the assumed variance of the random measurement error in that variable and, the attributes input (the name of the substitute variable) and call (the matched call).

Author(s)

Linda Nab, <l.nab@lumc.nl>

Examples

```
## random measurement error in a covariate:
# internal covariate-validation study
data(icvs)
with(icvs, MeasErrorRandom(X_star, variance = 0.25))
```

`mecor` *mecor: a Measurement Error Correction Package*

Description

mecor provides correction methods for measurement error in a continuous covariate or outcome in linear regression models with a continuous outcome

Usage

```
mecor(formula, data, method = "standard", B = 0)
```

Arguments

formula	an object of class formula (or one that is coerced to that class): a symbolic description of the regression model containing a MeasError, MeasErrorExt or MeasErrorRandom object in one of the covariates or the outcome.
data	a data.frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model specified in formula.
method	a character string indicating the method used to correct for the measurement error, either "standard" (regression calibration for covariate measurement error and method of moments for outcome measurement error), "efficient" (efficient regression calibration for covariate measurement error and efficient method of moments for outcome measurement error), "valregcal" (validation regression calibration) or "mle" (maximum likelihood estimation). Defaults to "standard".
B	number of bootstrap samples, defaults to 0.

Value

mecor returns an object of class "mecor".

An object of class mecor is a list containing the following components:

corfit	a list containing the corrected fit, including the coefficients of the corrected fit (coef) and the variance–covariance matrix of the coefficients of the corrected fit obtained by the delta method (vcov), and more depending on the method used.
uncorfit	an lm.fit object of the uncorrected fit.

Author(s)

Linda Nab, <l.nab@lumc.nl>

References

L. Nab, R.H.H. Groenwold, P.M.J. Welsing, and M. van Smeden. Measurement error in continuous endpoints in randomised trials: problems and solutions

L. Nab, M. van Smeden, R.H. Keogh, and R.H.H. Groenwold. mecor: an R package for measurement error correction in linear models with continuous outcomes

Examples

```
## measurement error in a covariate:
# internal covariate-validation study
data(icvs)
out <-
mecor(Y ~ MeasError(X_star, reference = X) + Z,
      data = icvs,
      method = "standard",
      B = 999)
# replicates study
data(rs)
mecor(Y ~ MeasError(X_star_1, replicate = cbind(X_star_2, X_star_3)) + Z1 + Z2,
```

```
      data = rs,
      method = "mle")
# covariate-calibration study
data(ccs)
mecor(Y ~ MeasError(X_star, replicate = cbind(X_star_1, X_star_2)) + Z,
      data = ccs,
      method = "efficient")
# external covariate-validation study
data(ecvs)
calmod_fit <- lm(X ~ X_star + Z, data = ecvs)
data(icvs) # suppose reference X is not available
mecor(Y ~ MeasErrorExt(X_star, model = calmod_fit) + Z,
      data = icvs)
# sensitivity analyses
data(icvs) # suppose reference X is not available
# guesstimate the coefficients of the calibration model:
mecor(Y ~ MeasErrorExt(X_star, model = list(coef = c(0, 0.9, 0.2))) + Z,
      data = icvs)
# assume random measurement error in X_star of magnitude 0.25:
mecor(Y ~ MeasErrorRandom(X_star, variance = 0.25) + Z,
      data = icvs)
data(rs) # suppose replicates X_star_2 and X_star_2 are not available
mecor(Y ~ MeasErrorRandom(X_star_1, variance = 0.25) + Z1 + Z2,
      data = rs)

## measurement error in the outcome:
# internal outcome-validation study
data(iovs)
mecor(MeasError(Y_star, reference = Y) ~ X + Z,
      data = iovs,
      method = "standard")
# external outcome-validation study
data(eovs)
memod_fit <- lm(Y_star ~ Y, data = eovs)
data(iovs) # suppose reference Y is not available
mecor(MeasErrorExt(Y_star, model = memod_fit) ~ X + Z,
      data = iovs,
      method = "standard")
# sensitivity analyses
data(iovs) # suppose reference Y is not available
# guesstimate the coefficients of the measurement error model:
mecor(MeasErrorExt(Y_star, model = list(coef = c(0, 0.5))) ~ X + Z,
      data = iovs,
      method = "standard")

## differential measurement error in the outcome:
# internal outcome-validation study
data(iovs_diff)
mecor(MeasError(Y_star, reference = Y, differential = X) ~ X,
      data = iovs_diff,
      method = "standard")
# sensitivity analysis
data(iovs_diff) # suppose reference Y is not available
```

```
# guesstimate the coefficients of the measurement error model:
mecor(MeasErrorExt(Y_star, model = list(coef = c(0, 0.5, 1, 1))) ~ X,
      data = iovs_diff,
      method = "standard")
```

`rs` *Replicates Study*

Description

A simulated dataset containing 1000 observations of the outcome Y, three replicate measures of the error-prone exposure X1_star, X2_star and X3_star and two covariates Z1 and Z2.

Usage

```
rs
```

Format

A data frame with 1000 rows and 6 variables:

Y outcome, continuous

X_star_1 first replicate of error-prone exposure, continuous

X_star_2 second replicate of error-prone exposure, continuous

X_star_3 third replicate of error-prone exposure, continuous

Z1 covariate, continuous

Z2 covariate, continuous

Examples

```
data("rs", package = "mecor")
```

`sim` *Simulated dataset for the ipwm function*

Description

A simulated dataset containing 5000 observations of the covariates L1-L10, the true exposure A and true outcome Y, and the misclassified exposure B and misclassified outcome Z.

Usage

```
sim
```

Format

A data frame with 5000 rows and 14 variables:

L1 covariate, binary

L2 covariate, continuous

L3 covariate, binary

L4 covariate, continuous

L5 covariate, binary

L6 covariate, binary

L7 covariate, continuous

L8 covariate, binary

L9 covariate, binary

L10 covariate, continuous

A exposure, binary

Y outcome, binary

B misclassified exposure, binary

Z misclassified outcome, binary

Examples

```
data("sim", package = "mecor")
```

---

summary.mecor *Summarizing Measurement Error Correction*

Description

summary method for class "mecor"

Usage

```
## S3 method for class 'mecor'
summary(object, alpha = 0.05, zerovar = FALSE, fieller = FALSE, ...)
```

Arguments

object	an object of class "mecor", a result of a call to mecor.
alpha	probability of obtaining a type II error.
zerovar	a boolean indicating whether standard errors and confidence intervals using the zerovariance method must be added to the summary object.
fieller	a boolean indicating whether confidence intervals using the fieller method must be added to the summary object.
...	additional arguments affecting the summary produced

Value

The function `summary.mecor` returns a list of summary statistics of the fitted corrected model and fitted uncorrected model.

`call`	the matched call
`c`	summary of the corrected fit
`uc`	summary of the uncorrected fit
`B`	number of bootstrap replicates used
`alpha`	alpha level used

See Also

The model fitting function mecor, summary

Examples

```
## measurement error in a covariate:
# internal covariate-validation study
data(icvs)
mecor_fit <- mecor(Y ~ MeasError(X_star, reference = X) + Z,
                   data = icvs,
                   method = "standard")
summary(mecor_fit)
summary(mecor_fit, zerovar = TRUE, fieller = TRUE)
summary(mecor_fit, alpha = 0.10)
```

Index