# Package ‘misclass’

March 1, 2022

**Title** Adjust for Misclassification of an Exposure Variable in Pooled Data

**Version** 0.0.0.9000

**Description** Adjust for misclassification of an exposure variable in pooled data, such as in an individual participant data meta-analysis (IPDMA). Using Bayesian misclassification models, the potential misclassification of an exposure is accounted for, and the uncertainty is propagated to the variance and CI. Modeling of an single data set (without clustering) is also possible. The misclassification model is fitted using misclass(). This package is intended to make using a misclassification model easier. Therefore, it is not entirely comprehensive, but it is designed to be flexible. The functions make.inits(), make.model() and make.monitors() can be used separately to tweak the model, but are otherwise called by misclass().

**Imports** coda, runjags, stats

**Depends** R (>= 4.0.0)

**Suggests** testthat (>= 3.1.0), mvtnorm

**License** GPL-3 + file LICENSE

**Encoding** UTF-8

**Date** 2021

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.1.2

**NeedsCompilation** no

**Author** Valentijn de Jong [aut, cre] (<`https://orcid.org/0000-0001-9921-3468`>)

**Maintainer** Valentijn de Jong <`Valentijn.M.T.de.Jong@gmail.com`>

## R topics documented:

## `coef.misclass` *Extract model coefficients*

### Description

Extract model coefficients from a misclass object.

### Usage

```
## S3 method for class 'misclass'
coef(object, parm, method = "Median", ...)
```

### Arguments

`object` a misclass object

`parm` parameter for which coefficients are required. Defaults to all

`method` Method for estimating the center of the posterior distribution. Defaults to "Median", the alternative is "Mean"

`...` ignored

## `confint.misclass` *Credible intervals*

### Description

Credible intervals for a misclass object.

### Usage

```
## S3 method for class 'misclass'
confint(object, parm, level = 0.95, ...)
```

### Arguments

`object` a misclass object

`parm` vector for names of parameters for which credibile intervals are requested. Defaults to all.

`level` the credible interval required

`...` ignored

### Details

Unlike the generic function `confint`, this function produces credible intervals, not confidence intervals. The intervals are estimated using the `quantile` of the mcmc samples.

## `extend.misclass` *Extend the mcmc chain of a misclass object*

### Description

When the `misclass` object contains too few samples, this function can extend the MCMC chain. By default, it doubles the total amount of samples used for inference, but does not perform another warmup or adaptation.

### Usage

```
extend.misclass(
  object,
  sample = object$jags.options$sample,
  warmup = 0,
  adapt = 0
)
```

### Arguments

| | |
|---|---|
| `object` | a `misclass` object |
| `sample` | Number of samples by which the mcmc chain(s) should be extended, defaults to same as original object (i.e. the total of samples is doubled). |
| `warmup` | Number of additional warmup iterations for the extension of the chain, defaults to 0. |
| `adapt` | Number of additional adaptation iterations of the mcmc sampler, defaults to 0. |

### Details

see `extend.jags` from the runjags package for details.

## `make.inits` *Make initial values for multiple chains*

### Description

Initial values for multiple JAGS chains by the misclass function.

### Usage

```
make.inits(
  J,
  n.z,
  n.chains,
  monitor,
  coef.init.limit = 2,
  tau2.init.min = 1e-04,
  tau2.init.max = 2
)
```

### Arguments

`J` Number of clusters

`n.z` Number of covariates (i.e. excluding the exposure)

`n.chains` Number of MCMC chains

`monitor` Names of parameters for which to generate initial values

`coef.init.limit` For coefficients, initial values are generated between -coef.init.limit and coef.init.limit.

`tau2.init.min` Minimum for initial values of tau2 heterogeneity parameters.

`tau2.init.max` Maximum for initial values of tau2 heterogeneity parameters.

### Value

List of vectors with named initial values.

### Examples

```
make.inits(
  J = 5,
  n.z = 2,
  n.chains = 3,
  monitor = c("eta.0.0", "phi.y.0", "tau2.beta.x.j")
)
```

---

## make.model *Make a misclassification model for JAGS*

---

### Description

Make a misclassification model that runjags/JAGS can read. The model can have covariate effects and random intercepts for all submodels and also random slopes for the outcome submodel. To view the model, save it to a file and open it (see examples).

### Usage

```
make.model(
 measurement = list(stratified.differential = FALSE, additive.differential = FALSE,
    covariate = TRUE, intercept.random = FALSE),
  exposure = list(covariate = TRUE, intercept.random = FALSE),
 outcome = list(covariate = TRUE, intercept.random = FALSE, effects.random = FALSE),
  priors = list(coef.precision = 0.1, heterogeneity.precision = 5)
)
```

### Arguments

`measurement` List for measurement submodel, containing: `stratified.differential` to define a stratified differential misclassification model (i.e. the measurement submodel is stratified by values of `y`); `additive.differential` to define a differential misclassification model with additive terms for `y` in the measurement submodel (i.e. linear on the log-odds scale); `covariate` to allow for covariate effects (for z); and `intercept.random` to allow for random intercepts in the measurement submodel. All may be abbreviated to the first letter.

`exposure` List for exposure submodel, containing: `covariate` to allow for covariate effects (for z) in the exposure submodel; and `intercept.random` to allow for random intercepts in the exposure submodel. All may be abbreviated to the first letter.

`outcome` List for outcome submodel, containing: `covariate` to allow for covariate effects (for z) in the outcome submodel; `intercept.random` to allow for random intercepts in the outcome submodel; and `effects.random` to allow for random effects (random slopes) in the outcome submodel. All may be abbreviated to the first letter.

`priors` List for priors, containing `coef.precision`: the value for the precision of the priors for the coefficients, and `heterogeneity.precision`: the value for the precision of the priors for the heterogeneity parameters.

## Details

`stratified.differential` Allows for completely different submodels for different values of y (and x), and thereby overrides `additive.differential`.

The output can be saved as .R or .txt file, making it easier to read, and allowing it to be run with rjags as well. See the examples.

## Value

a JAGS model, as a character string that `run.jags` from the `runjags` package can read.

## Examples

```
# Default: Nondifferential error, covariate effects but no random effects.
m <- make.model()

# Random intercepts. Note, not the whole list needs to be entered, only the
# arguments that are to be changed.
m <- make.model(measurement = list(intercept.random = TRUE),
exposure = list(intercept.random = TRUE),
outcome = list(intercept.random = TRUE))

# Random effects. Note that options of the lists can be abbreviated to the
# first letter.
m <- make.model(measurement = list(i = TRUE),
exposure = list(i = TRUE),
outcome = list(i = TRUE, e = TRUE))

# Differential misclassification
m <- make.model(measurement = list(s = TRUE))
m <- make.model(measurement = list(a = TRUE))

## Not run:
write(m, file = "jags_script.R")
file.edit("jags_script.R")
## End(Not run)
```

make.monitors *Make parameters for monitoring*

## Description

Make parameters for monitoring the JAGS object in misclass.

## Usage

```
make.monitors(
  measurement = list(stratified.differential = FALSE, additive.differential = FALSE,
    covariate = TRUE, intercept.random = FALSE),
   exposure = list(covariate = TRUE, intercept.random = FALSE),
  outcome = list(intercept.random = FALSE, slope.random = FALSE, covariate = TRUE),
   monitor.re = FALSE
)
```

## Arguments

`measurement` List for measurement submodel, containing: `stratified.differential` to define a stratified differential misclassification model (i.e. the measurement submodel is stratified by values of y); `additive.differential` to define a differential misclassification model with additive terms for y in the measurement submodel (i.e. linear on the log-odds scale); `covariate` to allow for covariate effects (for z); and `intercept.random` to allow for random intercepts in the measurement submodel. All may be abbreviated to the first letter.

`exposure` List for exposure submodel, containing: `covariate` to allow for covariate effects (for z) in the exposure submodel; and `intercept.random` to allow for random intercepts in the exposure submodel. All may be abbreviated to the first letter.

`outcome` List for outcome submodel, containing: `covariate` to allow for covariate effects (for z) in the outcome submodel; `intercept.random` to allow for random intercepts in the outcome submodel; and `effects.random` to allow for random effects (random slopes) in the outcome submodel. All may be abbreviated to the first letter.

`monitor.re` Should the estimated random effects per clusters be monitored? Defaults to FALSE, as this produces a lot of output when many clusters are present.

## Value

Returns a List of parameter names

## Examples

```
# Nondifferential model, with no random effects or random intercepts.
m <- make.monitors()

# Differential misclassification model with random intercepts, random effects,
# fixed covariate effects
m <- make.monitors(measurement = list(stratified.differential = TRUE,
                                      additive.differential   = FALSE,
                                 covariate              = TRUE,
                                 intercept.random       = TRUE),
                  exposure    = list(covariate              = TRUE,
                                 intercept.random       = TRUE),
                  outcome     = list(intercept.random       = TRUE,
                                 slope.random           = TRUE,
                                 covariate              = TRUE))
# Random intercepts (abbreviated to i), and covariate effects (which default
# to TRUE) for all three submodel.
m <- make.monitors(measurement = list(i = TRUE),
                   exposure    = list(i = TRUE),
                   outcome     = list(i = TRUE))
```

## misclass *Misclassification model for clustered binary data*

### Description

Misclassification model for clustered individual participant data with binary outcomes (endpoints), continuous or binary covariates and a binary exposure (or intervention) that is possibly misclassified in some studies.

### Usage

```
misclass(
  x,
  s,
  y,
  z,
  j,
  n.chains = 2,
  adapt = 1000,
  warmup = 5000,
  sample = 10000,
  thin = 1,
  psrf.max = 1.05,
  monitor.re = FALSE,
  monitor = NULL,
  inits = NULL,
  inits.options = NULL,
  measurement = NULL,
  exposure = NULL,
  outcome = NULL,
  priors = NULL,
  model = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| x | Gold standard measurement of the binary exposure, possibly missing. |

`s` Surrogate measurement of the binary exposure. Potentially misclassified.

`y` Binary outcome y. Possibly missing.

`z` Matrix or a single vector of covariates. If a matrix, each column should be one covariate, each row an observation/participant/patient.

`j` Indicator variable for the cluster (country, center, data set) the observation belongs to. If missing, all samples are assumed to belong to the same cluster.

`n.chains` Number of MCMC chains

`adapt` Number of iterations for the sampler to adapt before running the warmup and final analysis.

`warmup` Number of iterations for the sampler to find the posterior, before running the final analysis (also referred to as burnin).

`sample` Number of iterations for final analysis.

`thin` thinning interval. Values > 1 can be used to discard samples to reduce autocorrelation.

`psrf.max` maximum value for the psrf and mpsrf, to automatically to flag non-convergence quantified by `gelman.diag` from the `coda` package..

`monitor.re` Should the estimated random effects per cluster be monitored?

`monitor` Optional vector for setting monitors. Defaults to all fixed effects, and standard deviations for random effects/intercepts. Include include all random by setting `monitor.re` to `TRUE`.

`inits` Optional initial values. If left `NULL` separate initial values are generated for each chain, using `inits.options`

`inits.options` List of: `coef.init.limit` For coefficients, initial values are generated between coef.init.limit and coef.init.limit; `tau2.init.min` Minimum for initial values of tau2 heterogeneity parameters; and `tau2.init.max` Maximum for initial values of tau2 heterogeneity

`measurement` List for measurement submodel, containing: `stratified.differential` to define a stratified differential misclassification model (i.e. the measurement submodel is stratified by values of `y`); `additive.differential` to define a differential misclassification model with additive terms for `y` in the measurement submodel (i.e. linear on the log-odds scale); `covariate` to allow for covariate effects (for z); and `intercept.random` to allow for random intercepts in the measurement submodel. All may be abbreviated to the first letter.

`exposure` List for exposure submodel, containing: `covariate` to allow for covariate effects (for z) in the exposure submodel; and `intercept.random` to allow for random intercepts in the exposure submodel. All may be abbreviated to the first letter.

`outcome` List for outcome submodel, containing: `covariate` to allow for covariate effects (for z) in the outcome submodel; `intercept.random` to allow for random intercepts in the outcome submodel; and `effects.random` to allow for random effects (random slopes) in the outcome submodel. All may be abbreviated to the first letter.

`priors` List for priors, containing `coef.precision`: the value for the precision of the priors for the coefficients, and `heterogeneity.precision`: the value for the precision of the priors for the heterogeneity parameters.

`model` Optional JAGS model. Overrides `measurement`, `exposure`, `outcome`, and `priors`,

`...` Optional parameters for `run.jags` from the `runjags` package.

### Details

This function (and the rest of the package) is intended to make using a misclassification model easier. Therefore, it is not entirely comprehensive. However, the package is designed to be flexible, so that you can use partial outputs (the JAGS model, monitors and inits), if you would like to tweak the JAGS model.

For the coefficients and random intercepts/effects normal priors are used. For the variances of the heterogeneity across clusters, half-normal priors are used.

## print.misclass *Print a misclass object*

### Description

Print a misclass object

### Usage

```
## S3 method for class 'misclass'
print(x, digits = 3, vars = NA, ...)
```

### Arguments

| | |
|---|---|
| x | A misclass object |
| digits | Number of digits used for rounding |
| vars | Names of variables to print. Ignored for models with few variables. Defaults to all variables (which may be undesirable for models with many monitored variables). |
| ... | Arguments passed to `add.summary` from the `runjags` package, for models with many monitors. |

## sample.misclass *Sample clustered binary data with misclassified exposure*

### Description

Sample clustered binary data with misclassified exposure, correctly classified exposure, covariates and outcome. Random intercepts for outcome, measurementand exposure model. Random effects for outcome model.

## Usage

```
sample.misclass(
  n = 50,
  J = 1,
  beta.0.0 = NULL,
  tau.beta.0.j = 0,
  beta.x.0 = log(2),
  tau.beta.x.j = 0,
  beta.z.0 = log(2),
  tau.z.mu = 1/2,
  cov = 1/4,
  gamma.0.0 = 0,
  tau.gamma.0.j = 0,
  gamma.z.0 = 1/2,
  lambda.0.0 = 3,
  tau.lambda.0.j = 1,
  lambda.z.0 = -2,
  phi.0.0 = -3,
  tau.phi.0.j = 1,
  phi.z.0 = 2,
  colnames = NULL,
  center.z = TRUE
)
```

## Arguments

| | |
|---|---|
| `n` | sample size per cluster |
| `J` | number of clusters |
| `beta.0.0` | intercept in the outcome model. if NULL, the incidence is fixed so that mean(y) converges to 1/2. |
| `tau.beta.0.j` | variance of the random intercept in the outcome model |
| `beta.x.0` | coefficient for x in the outcome model |
| `tau.beta.x.j` | variance of random effect for x in the outcome model |
| `beta.z.0` | log odds ratios for covariates. |
| `tau.z.mu` | SD of the mean of the z covariates across studies |
| `cov` | Covariance for continuous covariates |
| `gamma.0.0` | intercept in the exposure model |
| `tau.gamma.0.j` | variance of the random intercept in the exposure model |
| `gamma.z.0` | coefficients for z in the exposure model |
| `lambda.0.0` | intercept in the measurement model, if x = 1 |
| `tau.lambda.0.j` | variance of the random intercept in the measurement model, if x = 1 |
| `lambda.z.0` | coefficients in the measurement model, if x = 1 |
| `phi.0.0` | intercept in the measurement model, if x = 0 |
| `tau.phi.0.j` | variance of the random intercept in the measurement model, if x = 0 |
| `phi.z.0` | coefficients in the measurement model, if x = 0 |
| `colnames` | optional column names |
| `center.z` | should covariates be centered? (post hoc) |

## Value

A data.frame, including the mismeasured and correctly measured variables.

## Examples

```
d <- sample.misclass(gamma.z.0 = c(1/2, 1/2),
lambda.z.0 = c(-1, -1),
phi.z.0 = c(1, 1),
beta.z.0 = c(1, 1))
```

# Index