# Package 'inlamemi'

May 8, 2026

**Type** Package

**Title** Missing Data and Measurement Error Modelling in INLA

**Version** 1.1.0

**Description** Facilitates fitting measurement error and missing data imputation models using integrated nested Laplace approximations, according to the method described in Skarstein, Martino and Muff (2023) <doi:10.1002/bimj.202300078>. See Skarstein and Muff (2024) <doi:10.48550/arXiv.2406.08172> fo tails on using the package.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**VignetteBuilder** knitr

**RoxygenNote** 7.3.1

**Suggests** INLA, knitr, testthat (>= 3.0.0), tibble, rmarkdown, spelling

**Additional_repositories** https://inla.r-inla-download.org/R/stable/

**Config/testthat/edition** 3

**Imports** dplyr, ggplot2, rlang, stats, methods, scales

**Depends** R (>= 2.10)

**URL** https://emmaskarstein.github.io/inlamemi/,
https://github.com/emmaSkarstein/inlamemi

**Language** en-US

**NeedsCompilation** no

**Author** Emma Skarstein [cre, aut, cph] (ORCID:
<https://orcid.org/0000-0002-6347-855X>),
Stefanie Muff [aut] (ORCID: <https://orcid.org/0000-0002-8425-0757>)

**Maintainer** Emma Skarstein <emma@skarstein.no>

**Repository** CRAN

**Date/Publication** 2024-10-31 16:10:02 UTC

# Contents



---

`coef.inlamemi` *Extract model coefficients*

---

## Description

Extract model coefficients

## Usage

```
## S3 method for class 'inlamemi'
coef(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | object of class ‘inlamemi‘ |
| `...` | other arguments |

`fit_inlamemi` *Fit model for measurement error and missing data in INLA*

## Description

A wrapper function around "INLA::inla()", providing the necessary structure to fit the hierarchical measurement error model that adjusts coefficient estimates to account for biases due to measurement error and missing data.

## Usage

```
fit_inlamemi(
  formula_moi,
  formula_imp = NULL,
  formula_mis = NULL,
  family_moi,
  data,
  error_type = "classical",
  error_variable = NULL,
  repeated_observations = FALSE,
  classical_error_scaling = NULL,
  prior.prec.moi = NULL,
  prior.prec.berkson = NULL,
  prior.prec.classical = NULL,
  prior.prec.imp = NULL,
  prior.beta.error = NULL,
  prior.gamma.error = NULL,
  initial.prec.moi = NULL,
  initial.prec.berkson = NULL,
  initial.prec.classical = NULL,
  initial.prec.imp = NULL,
  control.family.moi = NULL,
  control.family.berkson = NULL,
  control.family.classical = NULL,
  control.family.imp = NULL,
  control.family = NULL,
  control.predictor = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `formula_moi` | an object of class "formula", describing the main model to be fitted. |
| `formula_imp` | an object of class "formula", describing the imputation model for the mismeasured and/or missing observations. |

`formula_mis` an object of class "formula", describing the missingness model. Does not need to have a response variable, since this will always be a binary missingness indicator.

`family_moi` a string indicating the likelihood family for the model of interest (the main model).

`data` an object of class data.frame or list containing the variables in the model.

`error_type` type of error (one or more of "classical", "berkson", "missing")

`error_variable` character vector with the name(s) of the variable(s) with error.

`repeated_observations`
Does the variable with measurement error and/or missingness have repeated observations? If so, set this to "TRUE". In that case, when specifying the formula, use the name of the variable without any numbers, but when specifying the data, make sure that the repeated measurements end in a number, i.e "sbp1" and "sbp2".

`classical_error_scaling`
can be specified if the classical measurement error varies across observations. Must be a vector of the same length as the data.

`prior.prec.moi` a string containing the parameters for the prior for the precision of the residual term for the model of interest.

`prior.prec.berkson`
a string containing the parameters for the prior for the precision of the error term for the Berkson error model.

`prior.prec.classical`
a string containing the parameters for the prior for the precision of the error term for the classical error model.

`prior.prec.imp` a string containing the parameters for the precision of the latent variable x, which is the variable being described in the imputation model.

`prior.beta.error`
parameters for the Gaussian prior for the coefficient of the error prone variable.

`prior.gamma.error`
parameters for the Gaussian prior for the coefficient of the variable with missingness in the missingness model.

`initial.prec.moi`
the initial value for the precision of the residual term for the model of interest.

`initial.prec.berkson`
the initial value for the precision of the residual term for the Berkson error term.

`initial.prec.classical`
the initial value for the precision of the residual term for the classical error term.

`initial.prec.imp`
the initial value for the precision of the residual term for the latent variable r.

`control.family.moi`
control.family component for model of interest. Can be specified here using the inla syntax instead of passing the "prior.prec..." and "initial.prec..." arguments, or in the cases when other hyperparameters are needed for the model of interest, see for instance survival models.

`control.family.berkson`
: control.family component Berkson model. Can be specified here using the inla syntax instead of passing the "prior.prec..." and "initial.prec..." arguments. Useful in the cases when more flexibility is needed, for instance if one wants to specify a different prior distribution than Gamma.

`control.family.classical`
: control.family component for classical model. Can be specified here using the inla syntax instead of passing the "prior.prec..." and "initial.prec..." arguments. Useful in the cases when more flexibility is needed, for instance if one wants to specify a different prior distribution than Gamma.

`control.family.imp`
: control.family component for imputation model. Can be specified here using the inla syntax instead of passing the "prior.prec..." and "initial.prec..." arguments. Useful in the cases when more flexibility is needed, for instance if one wants to specify a different prior distribution than Gamma.

`control.family`
: control.family for use in inla (can be provided directly instead of passing the "prior.prec...." and "initial.prec..." arguments. If this is specified, any other "control.family..." or "prior.prec..." arguments provided will be ignored.

`control.predictor`
: control.predictor for use in inla.

`...`
: other arguments to pass to ‘inla‘.

## Value

An object of class `inlamemi`.

## Examples

```
# Fit the model
simple_model <- fit_inlamemi(data = simple_data,
                           formula_moi = y ~ x + z,
                           formula_imp = x ~ z,
                           family_moi = "gaussian",
                           error_type = c("berkson", "classical"),
                           error_variable = "x",
                           prior.prec.moi = c(10, 9),
                           prior.prec.berkson = c(10, 9),
                           prior.prec.classical = c(10, 9),
                           prior.prec.imp = c(10, 9),
                           prior.beta.error = c(0, 1/1000),
                           initial.prec.moi = 1,
                           initial.prec.berkson = 1,
                           initial.prec.classical = 1,
                           initial.prec.imp = 1)
```

framingham *Framingham heart study data*

## Description

A data set with observations of heart disease status systolic blood pressure (SBP) and smoking status.

## Usage

```
framingham
```

## Format

## 'framingham' A data frame with 641 rows and 4 columns:

**disease** A binary response, 1 if heart disease, 0 otherwise

**sbp1** log(SBP - 50) at examination 1 (centered)

**sbp2** log(SBP - 50) at examination 2 (centered)

**smoking** Smoking status, 1 if smoking, 0 otherwise.

## Source

MacMahon et al. (1990) <https://doi.org/10.1016/0140-6736(90)90878-9>

get_coef_imp *Extract coefficients for the imputation model (IMP)*

## Description

Extract coefficients for the imputation model (IMP)

## Usage

```
get_coef_imp(inlamemi_model)
```

## Arguments

| | |
|---|---|
| inlamemi_model | object of class 'inlamemi' |

## Value

A data frame with a summary of the posterior marginals for the coefficients in the imputation model.

## get_coef_mis *Extract coefficients for the missingness model (MIS)*

### Description

Extract coefficients for the missingness model (MIS)

### Usage

```
get_coef_mis(inlamemi_model)
```

### Arguments

| | |
|---|---|
| `inlamemi_model` | object of class 'inlamemi' |

### Value

A data frame with a summary of the posterior marginals for the coefficients in the missingness model.

## get_coef_moi *Extract coefficients for the model of interest (MOI)*

### Description

Extract coefficients for the model of interest (MOI)

### Usage

```
get_coef_moi(inlamemi_model)
```

### Arguments

| | |
|---|---|
| `inlamemi_model` | object of class 'inlamemi' |

### Value

A data frame with a summary of the posterior marginals for the coefficients in the model of interest.

## get_imputed *Extract imputed values*

### Description

Extract imputed values

### Usage

```
get_imputed(inlamemi_model, error_variable)
```

### Arguments

| | |
|---|---|
| `inlamemi_model` | object of class ‘inlamemi‘ |
| `error_variable` | character string indicating the name of the error variable for which to extract the imputed values. |

### Value

A list of two objects: the posterior marginal distributions for each element of the imputed covariate, and a data frame giving a summary of these marginals.

## mar_data *Simulated data with observation missing at random (MAR)*

### Description

A simulated dataset to demonstrate how to set up a model in the case where there are two variables with measurement error.

### Usage

```
mar_data
```

### Format

## ‘mar_data‘ A data frame with 1000 rows and 5 columns:

**y** Response variable

**x** Observed value of covariate, with almost 20 percent missing

**x_true** Correct version of x, without missingness

**z1** Covariate correlated with x

**z2** Covariate correlated with the missingness of x

### Source

The dataset is simulated.

## nhanes_survival *Survival data with repeated systolic blood pressure measurements*

### Description

A dataset containing a repeated blood pressure measurement along with some other variables for participants in the Third National Health and Nutrition Survey (NHANES III), merged with data from the US National Death Index by Ruth H. Keogh and Jonathan Bartlett. For the illustration purposes in this package, we have left out observations where smoking status is missing.

### Usage

```
nhanes_survival
```

### Format

## 'nhanes_survival' A data frame with 3433 rows and 8 columns:

**sbp1** systolic blood pressure (standardized), first measurement

**sbp2** systolic blood pressure (standardized), second measurement

**sex** sex (0 = female, 1 = male)

**age** age (standardized)

**smoke** smoking status (0 = no, 1 = yes)

**diabetes** diabetes status (0 = no, 1 = yes)

**d** censoring status (0 = censored, 1 = observed death due to cardiovascular disease)

**t** time until death due to cardiovascular disease occurs

### Source

https://github.com/ruthkeogh/meas_error_handbook

## plot.inlamemi *Plot model summary*

### Description

Plot model summary

## Usage

```
## S3 method for class 'inlamemi'
plot(
  x,
  plot_moi = TRUE,
  plot_imp = TRUE,
  plot_mis = TRUE,
  plot_intercepts = TRUE,
  error_variable_highlight = FALSE,
  greek = FALSE,
  palette = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| x | the model returned from the fit_inlamemi function. |
| plot_moi | should the posterior mean for the coefficients of the model of interest be plotted? Defaults to TRUE. |
| plot_imp | should the posterior mean for the coefficients of the imputation model be plotted? Defaults to TRUE. |
| plot_mis | should the posterior mean for the coefficients of the missingness model be plotted? Defaults to TRUE. |
| plot_intercepts | should the posterior mean for the intercept(s) be plotted? Defaults to TRUE. |
| error_variable_highlight | should the coefficient(s) of the variable(s) with error be highlighted? (circled in black) Defaults to FALSE. |
| greek | make the coefficient names into greek letters with the covariate name as subscript. Defaults to FALSE. |
| palette | either a number (between 1 and 5), indicating the number of the color palette to be used, or a vector of the colors to be used. |
| ... | other arguments |

## Value

An object of class "ggplot2" that plots the posterior mean and 95 % credible interval for each coefficient in the model. The coefficients are colored to indicate if they belong to the main or imputation model, and the variable with error is also highlighted.

## Examples

```
# Fit the model
simple_model <- fit_inlamemi(data = simple_data,
                             formula_moi = y ~ x + z,
                             formula_imp = x ~ z,
                             family_moi = "gaussian",
```

```
                              error_type = c("berkson", "classical"),
                              prior.prec.moi = c(10, 9),
                              prior.prec.berkson = c(10, 9),
                              prior.prec.classical = c(10, 9),
                              prior.prec.imp = c(10, 9),
                              prior.beta.error = c(0, 1/1000),
                              initial.prec.moi = 1,
                              initial.prec.berkson = 1,
                              initial.prec.classical = 1,
                              initial.prec.imp = 1)

plot(simple_model)
```

---

`print.inlamemi` *Print method for inlamemi*

---

## Description

Print method for inlamemi

## Usage

```
## S3 method for class 'inlamemi'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | object of class ‘inlamemi‘. |
| `...` | other arguments. |

---

`show_data_structure` *Visualize the model data structure as matrices in LaTeX*

---

## Description

Visualize the model data structure as matrices in LaTeX

## Usage

```
show_data_structure(stack)
```

## Arguments

| | |
|---|---|
| `stack` | an object of class inla.stack returned from the function make_inlamemi_stacks, which describes the structure of the data for the measurement error and imputation model. |

## Value

A list containing data frames with the left hand side (response_df) and right hand side (effects_df), along with the latex code needed to visualize the matrices (matrix_string).

## Examples

```
stack <- make_inlamemi_stacks(data = simple_data,
                  formula_moi = y ~ x + z,
                  formula_imp = x ~ z,
                  error_type = "classical")
show_data_structure(stack)
```

---

# simple_data *Simple simulated data*

---

## Description

A simulated dataset to demonstrate how to model different types of measurement error and missing data using the 'inlamemi' package.

## Usage

```
simple_data
```

## Format

## 'simple_data' A data frame with 1000 rows and 4 columns:

**y** Response variable

**x** Covariate measured with error, both Berkson and classical error and missing observations

**x_true** Correct version of the covariate with error

**z** Error free covariate, correlated with x

## Source

The dataset is simulated.

summary.inlamemi *Summary method for inlamemi*

## Description

Takes a fitted ‘inlamemi‘ object produced by ‘fit_inlamemi‘ and produces a summary from it.

## Usage

```
## S3 method for class 'inlamemi'
summary(object, ...)

## S3 method for class 'summary.inlamemi'
print(x, ...)
```

## Arguments

| | |
|---|---|
| object | model of class ‘inlamemi‘. |
| ... | other arguments |
| x | object of class summary.inlamemi. |

## Value

‘summary.inlamemi‘ returns an object of class ‘summary.inlamemi‘, a list of components to print.

## Examples

```
# Fit the model
simple_model <- fit_inlamemi(data = simple_data,
                             formula_moi = y ~ x + z,
                             formula_imp = x ~ z,
                             family_moi = "gaussian",
                             error_type = c("berkson", "classical"),
                             prior.prec.moi = c(10, 9),
                             prior.prec.berkson = c(10, 9),
                             prior.prec.classical = c(10, 9),
                             prior.prec.imp = c(10, 9),
                             prior.beta.error = c(0, 1/1000),
                             initial.prec.moi = 1,
                             initial.prec.berkson = 1,
                             initial.prec.classical = 1,
                             initial.prec.imp = 1)

summary(simple_model)
```

`two_error_data` *Simulated data with two covariates with classical measurement error*

## Description

A simulated dataset to demonstrate how to set up a model in the case where there are two variables with measurement error.

## Usage

```
two_error_data
```

## Format

## 'two_error_data' A data frame with 1000 rows and 5 columns:

**y** Response variable

**x1** Covariate measured with classical error, correlated with z

**x2** Covariate measured with classical error

**x1_true** Correct version of x1

**x2_true** Correct version of x2

**z** Error free covariate, correlated with x1

## Source

The dataset is simulated.

# Index