

Package ‘JointAI’

February 12, 2020

Version 0.6.1

Title Joint Analysis and Imputation of Incomplete Data

Description Provides joint analysis and imputation of (generalized) linear and cumulative logit regression models, (generalized) linear and cumulative logit mixed models and parametric (Weibull) as well as Cox proportional hazards survival models with incomplete (covariate) data in the Bayesian framework.
The package performs some preprocessing of the data and creates a 'JAGS' model, which will then automatically be passed to 'JAGS' <<http://mcmc-jags.sourceforge.net>> with the help of the package 'rjags'.
It also provides summary and plotting functions for the output and allows the user to export imputed values.

URL <https://nerler.github.io/JointAI>

License GPL (>= 2)

Date 2020-02-12

BugReports <https://github.com/nerler/JointAI/issues>

LazyData TRUE

RoxygenNote 7.0.2

Depends rjags (>= 4-6)

Imports MASS, mcmcse, coda, rlang, foreach, doParallel

SystemRequirements JAGS (<http://mcmc-jags.sourceforge.net>)

Suggests knitr, rmarkdown, bookdown, foreign, ggplot2, ggpubr, survival, testthat

VignetteBuilder knitr

Encoding UTF-8

Language en-US

NeedsCompilation no

Author Nicole S. Erler [aut, cre] (<<https://orcid.org/0000-0002-9370-6832>>)

Maintainer Nicole S. Erler <n.erler@erasmusmc.nl>

Repository CRAN

Date/Publication 2020-02-12 09:30:02 UTC

R topics documented:

add_samples	2
default_hyperpars	4
densplot	6
get_MIdat	8
get_models	9
GR_crit	11
JointAI	12
JointAIObject	14
list_models	16
longDF	17
MC_error	18
md_pattern	20
model_imp	22
NHANES	33
parameters	34
plot.JointAI	35
plot_all	36
plot_imp_distr	37
predDF	38
predict.JointAI	39
residuals.JointAI	41
set_refcat	41
sharedParams	42
simLong	43
summary.JointAI	45
traceplot	47
wideDF	49
Index	51

add_samples

Continue sampling from an object of class JointAI

Description

This function allows to continue sampling from an existing object of class 'JointAI'.
 If the original sample was created using parallel computation, the separate 'jags' objects will be recompiled and sampling will again be performed in parallel.

Usage

```
add_samples(
  object,
  n.iter,
  add = TRUE,
  thin = NULL,
  monitor_params = NULL,
  progress.bar = "text",
  mess = TRUE
)
```

Arguments

object	object inheriting from class 'JointAI'
n.iter	the number of iterations of the MCMC chain (after adaptation; see also coda.samples)
add	logical; should the new MCMC samples be added to the existing samples or replace them? If samples are added, var.names is ignored.
thin	thinning interval (see window.mcmc)
monitor_params	named vector specifying which parameters should be monitored (see details)
progress.bar	character string specifying the type of progress bar. Possible values are "text", "gui", and "none" (see update). Note: when sampling is performed in parallel it is currently not possible to display a progress bar.
mess	logical; should messages be given? Default is TRUE. (Note: this applies only to messages given directly by JointAI .)

Note

When the thinning interval differs between the original model and the added samples, the resulting, new, 'JointAI' object does not yet record this information. Moreover, when `add_samples()` is used with `add = FALSE` the indices of iterations (for example given in the `summary()`) may not be correct.

See Also

[*_imp](#)

The vignette [Parameter Selection](#) contains some examples on how to specify the argument `monitor_params`.

Examples

```
# Example 1:
# Run an initial JointAI model:
mod <- lm_imp(y ~ C1 + C2, data = wideDF, n.iter = 100)

# Continue sampling:
mod_add <- add_samples(mod, n.iter = 200, add = TRUE)

# Example 2:
```

```
# Continue sampling, but additionally sample imputed values.
# Note: Setting different parameters to monitor than in the original model
# requires add = FALSE.
imps <- add_samples(mod, n.iter = 200, monitor_params = c("imps" = TRUE),
                    add = FALSE)
```

default_hyperpars *Get the default values for hyperparameters*

Description

This function returns a list of default values for the hyperparameters.

Usage

```
default_hyperpars()
```

Details

norm: hyperparameters for normal and lognormal models

mu_reg_norm	mean in the priors for regression coefficients
tau_reg_norm	precision in the priors for regression coefficients
shape_tau_norm	shape parameter in Gamma prior for precision of an imputed variable
rate_tau_norm	rate parameter in Gamma prior for precision of an imputed variable

gamma: hyperparameters for Gamma models

mu_reg_gamma	mean in the priors for regression coefficients
tau_reg_gamma	precision in the priors for regression coefficients
shape_tau_gamma	shape parameter in Gamma prior for precision of an imputed variable
rate_tau_gamma	rate parameter in Gamma prior for precision of an imputed variable

beta: hyperparameters for beta models

mu_reg_beta	mean in the priors for regression coefficients
tau_reg_beta	precision in the priors for regression coefficients
shape_tau_beta	shape parameter in Gamma prior for precision of imputed variable
rate_tau_beta	rate parameter in Gamma prior for precision of imputed variable

logit: hyperparameters for logistic models

mu_reg_logit	mean in the priors for regression coefficients
tau_reg_logit	precision in the priors for regression coefficients

probit: hyperparameters for probit models

mu_reg_logit	mean in the priors for regression coefficients
tau_reg_logit	precision in the priors for regression coefficients

multinomial: hyperparameters for multinomial models

mu_reg_multinomial	mean in the priors for regression coefficients
tau_reg_multinomial	precision in the priors for regression coefficients

ordinal: hyperparameters for ordinal models

mu_reg_ordinal	mean in the priors for regression coefficients
tau_reg_ordinal	precision in the priors for regression coefficients
mu_delta_ordinal	mean in the prior for the intercepts
tau_delta_ordinal	precision in the priors for the intercepts

Z: function creating hyperparameters for the random effects in mixed models, with output elements

RinvD	scale matrix in Wishart prior (*) for random effects covariance matrix
KinvD	degrees of freedom in Wishart prior for random effects covariance matrix
shape_diag_RinvD	shape parameter in Gamma prior for the diagonal elements of RinvD
rate_diag_RinvD	rate parameter in Gamma prior for the diagonal elements of RinvD

(*) when there is only one random effect a Gamma distribution is used instead of the Wishart and RinvD and KinvD are NULL

surv: parameters for survival models (parametric and proportional hazard)

mu_reg_surv	mean in the priors for regression coefficients
tau_reg_surv	precision in the priors for regression coefficients

coxph: parameters for Cox proportional hazards models

c	confidence in prior guess for the hazard function
r	failure rate per unit time
eps	time increment

Examples

```
default_hyperpars()

# To change the hyperparameters:
hyp <- default_hyperpars()
hyp$norm['rate_tau_norm'] <- 1e-3
mod <- lm_imp(y ~ C1 + C2 + B1, data = wideDF, hyperpars = hyp, mess = FALSE)
```

densplot

*Plot the posterior density from object of class JointAI***Description**

The function plots a set of densities (per chain and coefficient) from the MCMC sample of an object of class "JointAI".

Usage

```
densplot(object, ...)

## S3 method for class 'mcmc.list'
densplot(object, start = NULL, end = NULL, thin = NULL, ...)

## S3 method for class 'JointAI'
densplot(
  object,
  start = NULL,
  end = NULL,
  thin = NULL,
  subset = c(analysis_main = TRUE),
  exclude_chains = NULL,
  vlines = NULL,
  nrow = NULL,
  ncol = NULL,
  joined = FALSE,
  use_ggplot = FALSE,
  keep_aux = FALSE,
  warn = TRUE,
  mess = TRUE,
  ...
)
```

Arguments

object	object inheriting from class 'JointAI'
...	additional parameters passed to plot
start	the first iteration of interest (see window.mcmc)
end	the last iteration of interest (see window.mcmc)
thin	thinning interval (see window.mcmc)
subset	subset of parameters/variables/nodes (columns in the MCMC sample). Uses the same logic as the argument <code>monitor_params</code> in *_imp .
exclude_chains	optional vector of the index numbers of chains that should be excluded

vlines	list, where each element is a named list of parameters that can be passed to abline to create vertical lines. Each of the list elements needs to contain at least <code>v = <x location></code> , where <code><x location></code> is a vector of the same length as the number of plots (see examples).
nrow	optional number of rows and columns in the plot layout; automatically chosen if unspecified
ncol	optional number of rows and columns in the plot layout; automatically chosen if unspecified
joined	logical; should the chains be combined before plotting?
use_ggplot	logical; Should ggplot be used instead of the base graphics?
keep_aux	logical; Should constant effects of auxiliary variables be kept in the output?
warn	logical; should warnings be given? Default is TRUE. (Note: this applies only to warnings given directly by JointAI .)
mess	logical; should messages be given? Default is TRUE. (Note: this applies only to messages given directly by JointAI .)

See Also

The vignette [Parameter Selection](#) contains some examples how to specify the argument subset.

Examples

```
# fit a JointAI object:
mod <- lm_imp(y ~ C1 + C2 + M1, data = wideDF, n.iter = 100)

# Example 1: basic densityplot
densplot(mod)
densplot(mod, exclude_chains = 2)

# Example 2: use vlines to mark zero
densplot(mod, col = c("darkred", "darkblue", "darkgreen"),
          vlines = list(list(v = rep(0, nrow(summary(mod)$stats)),
                           col = grey(0.8))))

# Example 3: use vlines to visualize the posterior mean and 2.5% and 97.5% quantiles
densplot(mod, vlines = list(list(v = summary(mod)$stats[, "Mean"], lty = 1, lwd = 2),
                             list(v = summary(mod)$stats[, "2.5%"], lty = 2),
                             list(v = summary(mod)$stats[, "97.5%"], lty = 2)))

# Example 4: ggplot version
densplot(mod, use_ggplot = TRUE)

# Example 5: changing how the ggplot version looks (using standard ggplot syntax)
library(ggplot2)
```

```
densplot(mod, use_ggplot = TRUE) +
  xlab("value") +
  theme(legend.position = 'bottom') +
  scale_color_brewer(palette = 'Dark2', name = 'chain')
```

get_MIdat

Extract multiple imputed datasets from an object of class JointAI

Description

This function returns a dataset containing multiple imputed datasets stacked onto each other (i.e., long format; optionally including the original, incomplete data).

These data can be automatically exported to SPSS (i.e., a .txt file containing the data and a .sps file containing syntax to generate a .sav file). For the export function the **foreign** package needs to be installed.

Usage

```
get_MIdat(
  object,
  m = 10,
  include = TRUE,
  start = NULL,
  minspace = 50,
  seed = NULL,
  export_to_SPSS = FALSE,
  resdir = NULL,
  filename = NULL
)
```

Arguments

object	object inheriting from class 'JointAI'
m	number of imputed datasets
include	should the original, incomplete data be included? Default is TRUE.
start	the first iteration of interest (see window.mcmc)
minspace	minimum number of iterations between iterations chosen as imputed values.
seed	optional seed
export_to_SPSS	logical; should the completed data be exported to SPSS?
resdir	optional directory for results (if unspecified and export_to_SPSS = TRUE the current working directory is used)
filename	optional file name (without ending; if unspecified and export_to_SPSS = TRUE a name is generated automatically)

Value

A dataframe in which the original data (if `include = TRUE`) and the imputed datasets are stacked onto each other.

The variable `Imputation_` indexes the imputation, while `.rownr` links the rows to the rows of the original data. In cross-sectional datasets the variable `.id` is added as subject identifier.

Note

In order to be able to extract (multiple) imputed datasets the imputed values must have been monitored, i.e., `imps = TRUE` had to be specified in the argument `monitor_params` in `*_imp`.

See Also

[plot_imp_distr](#)

Examples

```
# fit a model and monitor the imputed values with monitor_params = c(imps = TRUE)
mod <- lm_imp(y ~ C1 + C2 + M2, data = wideDF, monitor_params = c(imps = TRUE), n.iter = 100)
```

```
# Example 1: without export to SPSS
MIs <- get_MIdat(mod, m = 3, seed = 123)
```

```
## Not run:
```

```
# Example 2: with export for SPSS (here: to the temporary directory "temp_dir")
temp_dir <- tempdir()
MIs <- get_MIdat(mod, m = 3, seed = 123, resdir = temp_dir,
                filename = "example_imputation",
                export_to_SPSS = TRUE)
```

```
## End(Not run)
```

get_models

Specify the default (imputation) model types

Description

Specify the default (imputation) model types

Usage

```
get_models(
  fixed,
  random = NULL,
  data,
```

```

    auxvars = NULL,
    no_model = NULL,
    models = NULL
  )

```

Arguments

fixed	a two sided formula describing the fixed-effects part of the model (see formula)
random	only for multi-level models: a one-sided formula of the form $\sim x_1 + \dots + x_n \mid g$, where $x_1 + \dots + x_n$ specifies the model for the random effects and g the grouping variable
data	a <code>data.frame</code>
auxvars	optional one-sided formula of variables that should be used as predictors in the imputation procedure (and will be imputed if necessary) but are not part of the analysis model
no_model	names of variables for which no model should be specified. Note that this is only possible for completely observed variables and implies the assumptions of independence between the excluded variable and the incomplete variables.
models	optional named vector specifying the types of models for (incomplete) covariates. This arguments replaces the argument <code>meth</code> used in earlier versions. If <code>NULL</code> (default) models will be determined automatically based on the class of the respective columns of data.

Value

`get_models()` returns a list of two vectors named `models` and `meth`.
`models` is a named vector containing the names of covariates that either have missing values and/or are longitudinal (level-1) covariates and the corresponding (imputation) models as well as models for variables for which the user has specified a model.
`meth` is a subset of `models` containing only the variables that have missing values.

Examples

```

get_models(y ~ C1 + C2 + B2 + O2 + M2, data = wideDF)

get_models(y ~ C1 + O2 + c2 + b1 + o2 + time, random = ~ 1 | id, data = longDF)

get_models(y ~ C1 + O2 + c2 + b1 + o2 + time, random = ~ 1 | id,
           no_model = 'time', data = longDF)

get_models(y ~ C1 + O2 + c2 + b1 + o2 + time, random = ~ 1 | id,
           no_model = 'time', data = longDF, models = c(C1 = 'norm'))

```

GR_crit

Gelman-Rubin criterion for convergence

Description

Calculates the Gelman-Rubin criterion for convergence (uses [gelman.diag](#) from package **coda**).

Usage

```
GR_crit(
  object,
  confidence = 0.95,
  transform = FALSE,
  autoburnin = TRUE,
  multivariate = TRUE,
  subset = NULL,
  exclude_chains = NULL,
  start = NULL,
  end = NULL,
  thin = NULL,
  warn = TRUE,
  mess = TRUE,
  ...
)
```

Arguments

object	object inheriting from class 'JointAI'
confidence	the coverage probability of the confidence interval for the potential scale reduction factor
transform	a logical flag indicating whether variables in x should be transformed to improve the normality of the distribution. If set to TRUE, a log transform or logit transform, as appropriate, will be applied.
autoburnin	a logical flag indicating whether only the second half of the series should be used in the computation. If set to TRUE (default) and start(x) is less than end(x)/2 then start of series will be adjusted so that only second half of series is used.
multivariate	a logical flag indicating whether the multivariate potential scale reduction factor should be calculated for multivariate chains
subset	subset of parameters/variables/nodes (columns in the MCMC sample). Uses the same logic as the argument monitor_params in *_imp.
exclude_chains	optional vector of the index numbers of chains that should be excluded
start	the first iteration of interest (see window.mcmc)
end	the last iteration of interest (see window.mcmc)

thin	thinning interval (see window.mcmc)
warn	logical; should warnings be given? Default is TRUE. (Note: this applies only to warnings given directly by JointAI .)
mess	logical; should messages be given? Default is TRUE. (Note: this applies only to messages given directly by JointAI .)
...	currently not used

References

Gelman, A and Rubin, DB (1992) Inference from iterative simulation using multiple sequences, *Statistical Science*, **7**, 457-511.

Brooks, SP. and Gelman, A. (1998) General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, **7**, 434-455.

See Also

The vignette [Parameter Selection](#) contains some examples how to specify the argument subset.

Examples

```
mod1 <- lm_imp(y ~ C1 + C2 + M2, data = wideDF, n.iter = 100)
GR_crit(mod1)
```

JointAI

JointAI: Joint Analysis and Imputation of Incomplete Data

Description

The **JointAI** package performs simultaneous imputation and inference for incomplete data using the Bayesian framework. Distributions of incomplete variables, conditional on other covariates, are specified automatically and modeled jointly with the analysis model. MCMC sampling is performed in '**JAGS**' via the R package **rjags**.

Main functions

The package has the following main functions that allow analysis in different settings:

- [lm_imp](#) for linear regression
- [glm_imp](#) for generalized linear regression
- [clm_imp](#) for (ordinal) cumulative logit models
- [lme_imp](#) for linear mixed models
- [glme_imp](#) for generalized linear mixed models
- [clmm_imp](#) for (ordinal) cumulative logit mixed models

- `survreg_imp` for parametric (Weibull) survival models
- `coxph_imp` for Cox proportional hazard models

As far as possible, the specification of these functions is analogue to the specification of their complete data versions `lm`, `glm`, `clm` (from the package **ordinal**), `lme` (from the package **nlme**), `clmm2` (from the package **ordinal**), `survreg` (from the package **survival**) and `coxph` (from the package **survival**).

Computations can be performed in parallel using the argument `parallel = TRUE`, the argument `ridge` allows the user to impose a ridge penalty on the regression coefficients of the analysis model, and hyperparameters can be changed via the argument `hyperpars` and the function `default_hyperpars`.

Results can be summarized and printed with `summary()`, `coef()` and `confint()`, and visualized using `traceplot()` or `densplot()`. The function `predict()` allows prediction (including credible intervals) from JointAI models.

Evaluation and export

Two criteria for evaluation of convergence and precision of the posterior estimate are available:

- `GR_crit` implements the Gelman-Rubin criterion ('potential scale reduction factor') for convergence
- `MC_error` calculates the Monte Carlo error to evaluate the precision of the MCMC sample

Imputed data can be extracted (and exported to SPSS) using `get_MIdat()`. The function `plot_imp_distr()` allows visual comparison of the distribution of observed and imputed values.

Other useful functions

- `parameters` and `list_models` to gain insight in the specified model
- `plot_all` and `md_pattern` to visualize the distribution of the data and the missing data pattern

Vignettes

The following vignettes are available

- *Minimal Example:*
A minimal example demonstrating the use of `lm_imp`, `summary.JointAI`, `traceplot` and `densplot`.
- *Visualizing Incomplete Data:*
Demonstrations of the options in `plot_all` (plotting histograms and barplots for all variables in the data) and `md_pattern` (plotting or printing the missing data pattern).
- *Model Specification:*
Explanation and demonstration of all parameters that are required or optional to specify the model structure in `lm_imp`, `glm_imp` and `lme_imp`. Among others, the functions `parameters`, `list_models`, `get_models` and `set_refcat` are used.
- *Parameter Selection:*
Examples on how to select the parameters/variables/nodes to follow using the argument `monitor_params` and the parameters/variables/nodes displayed in the `summary`, `traceplot`, `densplot` or when using `GR_crit` or `MC_error`.

- **MCMC Settings:**
Examples demonstrating how to set the arguments controlling settings of the MCMC sampling, i.e., `n.adapt`, `n.iter`, `n.chains`, `thin`, `inits`.
- **After Fitting:**
Examples on the use of functions to be applied after the model has been fitted, including `traceplot`, `densplot`, `summary`, `GR_crit`, `MC_error`, `predict`, `predDF` and `get_MIdat`.
- **Theoretical Background:**
Explanation of the statistical method implemented in **JointAI**.

References

- Nicole S. Erler, Dimitris Rizopoulos and Emmanuel M.E.H. Lesaffre (2019). JointAI: Joint Analysis and Imputation of Incomplete Data in R. *arXiv e-prints*, arXiv:1907.10867. URL <https://arxiv.org/abs/1907.10867>.
- Erler, N.S., Rizopoulos, D., Rosmalen, J., Jaddoe, V.W.V., Franco, O. H., & Lesaffre, E.M.E.H. (2016). Dealing with missing covariates in epidemiologic studies: A comparison between multiple imputation and a full Bayesian approach. *Statistics in Medicine*, 35(17), 2955-2974. doi: [10.1002/sim.6944](https://doi.org/10.1002/sim.6944)
- Erler, N.S., Rizopoulos D., Jaddoe, V.W.V., Franco, O.H. & Lesaffre, E.M.E.H. (2019). Bayesian imputation of time-varying covariates in linear mixed models. *Statistical Methods in Medical Research*, 28(2), 555–568. doi: [10.1177/0962280217730851](https://doi.org/10.1177/0962280217730851)

JointAIObject	<i>Fitted object of class 'JointAI'</i>
---------------	---

Description

An object returned by one of the main functions `*_imp`.

Value

<code>analysis_type</code>	<code>lm</code> , <code>glm</code> , <code>clm</code> , <code>lme</code> , <code>glme</code> , <code>clmm</code> , <code>survreg</code> or <code>coxph</code> with attributes <code>family</code> and <code>link</code>
<code>data</code>	the original (incomplete) dataset
<code>models</code>	named vector specifying the models used for longitudinal and incomplete covariates
<code>fixed</code>	supplied fixed effects formula
<code>random</code>	supplied random effects formula
<code>Mlist</code>	a list: containing the data, split up into <ul style="list-style-type: none"> • <code>outcome</code> (<code>y</code>) • event indicator for survival outcomes (<code>event</code>) • cross-sectional main effects (<code>Xc</code>) • cross-sectional interactions (<code>Xic</code>) • longitudinal main effects (<code>Xl</code>)

- longitudinal interactions (X_l)
- categorical cross-sectional incomplete variables (X_{cat})
- categorical longitudinal variables (X_{lcat})
- transformed cross-sectional variables (X_{trafo})
- transformed longitudinal variables (X_{ltrafo})
- random effects design matrix (Z)

and other important specifications:

- a list naming which columns of the above matrices are covariates in the analysis model (cols_{main})
- a list giving the names of the covariates in the analysis model per matrix (names_{main})
- specification for transformations (trafos)
- specification for hierarchical centering (hc_{list})
- reference values and dummies for categorical variables (refs)
- formula specifying auxiliary variables (auxvars)
- grouping specification (groups)
- the vector of variables to be scaled (scale_{vars})
- updated fixed effects structure (fixed2)
- the number of categories if the outcome of the analysis model is categorical (ncat)
- the number of subjects (N)
- whether posterior predictive checks are to be enabled ppc (not yet used)
- whether ridge shrinkage priors should be used for the regression coefficients of the analysis model (ridge)
- the number of random effects (nraneff)

K	matrix specifying the indices of the regression coefficients that are related to different parts of the model
K _{imp}	matrix specifying the indices of regression coefficients for the imputation models relating to different covariates
mcmc_settings	a list with elements modelfile name and path of JAGS model file n.chains number of MCMC chains n.adapt number of iterations in the adaptive phase n.iter number of iterations in the MCMC sample variable.names monitored nodes thin thinning of the MCMC sample inits a list containing the initial values that were passed to rjags parallel whether parallel sampling was used n.cores how many cores were used in parallel sampling
monitor_params	the list of parameter groups to be monitored
data_list	list with data that was passed to rjags
scale_pars	matrix with parameters used to center and scale the continuous variables

hyperpars	a list containing the values of the hyperparameters used
imp_par_list	a list with parameters used to write the imputation model syntax
model	JAGS model
sample	MCMC sample on the sampling scale (included only if keep_scaled_sample = TRUE)
MCMC	MCMC sample, scaled back to the scale of the data
time	the computational time used for the sampling (adaptive phase + sampling)
fitted.values	fitted (or predicted) values (if available)
residuals	residuals (if available)
call	the original call

list_models	<i>List covariate models</i>
-------------	------------------------------

Description

This function prints information on models specified for (incomplete) covariates in a JointAI object, including the model type, names of the parameters used and hyperparameters.

Usage

```
list_models(
  object,
  predvars = TRUE,
  regcoef = TRUE,
  otherpars = TRUE,
  priors = TRUE,
  refcat = TRUE
)
```

Arguments

object	object inheriting from class 'JointAI'
predvars	logical; should information on the predictor variables be printed? (default is TRUE)
regcoef	logical; should information on the regression coefficients be printed? (default is TRUE)
otherpars	logical; should information on other parameters be printed? (default is TRUE)
priors	logical; should information on the priors (and hyperparameters) be printed? (default is TRUE)
refcat	logical; should information on the reference category be printed? (default is TRUE)

Note

The models listed by this function are not the actual imputation models, but the conditional models that are part of the specification of the joint distribution. Briefly, the joint distribution is specified as a sequence of conditional models

$$p(y|x_1, x_2, x_3, \dots, \theta)p(x_1|x_2, x_3, \dots, \theta)p(x_2|x_3, \dots, \theta)\dots$$

The actual imputation models are the full conditional distributions $p(x_1|\cdot)$ derived from this joint distribution. Even though the conditional distributions do not contain the outcome and all other covariates in their linear predictor, outcome and other covariates are taken into account implicitly, since imputations are sampled from the full conditional distributions. For more details, see Erler et al. (2016) and Erler et al. (2019).

The function `list_models` prints information on the conditional distributions of the covariates (since they are what is specified; the full-conditionals are automatically derived within JAGS). The outcome is, thus, not part of the printed linear predictor, but is still included during imputation.

References

Erler, N.S., Rizopoulos, D., Rosmalen, J.V., Jaddoe, V.W., Franco, O.H., & Lesaffre, E.M.E.H. (2016). Dealing with missing covariates in epidemiologic studies: A comparison between multiple imputation and a full Bayesian approach. *Statistics in Medicine*, 35(17), 2955-2974.

Erler, N.S., Rizopoulos D. and Lesaffre E.M.E.H. (2019). JointAI: Joint Analysis and Imputation of Incomplete Data in R. *arXiv e-prints*, arXiv:1907.10867. URL <https://arxiv.org/abs/1907.10867>.

Examples

```
# (set n.adapt = 0 and n.iter = 0 to prevent MCMC sampling to save time)
mod1 <- lm_imp(y ~ C1 + C2 + M2 + O2 + B2, data = wideDF, n.adapt = 0, n.iter = 0, mess = FALSE)

list_models(mod1)
```

longDF

Longitudinal example dataset

Description

A simulated longitudinal dataset.

Usage

```
data(longDF)
```

Format

A simulated data frame with 329 rows and 21 variables with data from 100 subjects:

C1 continuous, complete baseline variable
C2 continuous, incomplete baseline variable
B1 binary, complete baseline variable
B2 binary, incomplete baseline variable
M1 unordered factor; complete baseline variable
M2 unordered factor; incomplete baseline variable
O1 ordered factor; complete baseline variable
O2 ordered factor; incomplete baseline variable
P1 count variable; complete baseline variable
P2 count variable; incomplete baseline variable
c1 continuous, complete longitudinal variable
c2 continuous incomplete longitudinal variable
b1 binary, complete longitudinal variable
b2 binary incomplete longitudinal variable
o1 ordered factor; complete longitudinal variable
o2 ordered factor; incomplete longitudinal variable
p1 count variable; complete longitudinal variable
p2 count variable; incomplete longitudinal variable
id id (grouping) variable
time continuous complete longitudinal variable
y continuous, longitudinal (outcome) variable

MC_error

Monte Carlo error

Description

Calculate, print and plot the Monte Carlo error of the samples from a JointAI model.

Usage

```
MC_error(
  x,
  subset = NULL,
  exclude_chains = NULL,
  start = NULL,
  end = NULL,
  thin = NULL,
```

```

    digits = 2,
    warn = TRUE,
    mess = TRUE,
    ...
)

## S3 method for class 'MCElist'
plot(x, data_scale = TRUE, plotpars = NULL, ablinepars = list(v = 0.05), ...)

```

Arguments

<code>x</code>	object inheriting from class 'JointAI'
<code>subset</code>	subset of parameters/variables/nodes (columns in the MCMC sample). Uses the same logic as the argument <code>monitor_params</code> in <code>*_imp</code> .
<code>exclude_chains</code>	optional vector of the index numbers of chains that should be excluded
<code>start</code>	the first iteration of interest (see <code>window.mcmc</code>)
<code>end</code>	the last iteration of interest (see <code>window.mcmc</code>)
<code>thin</code>	thinning interval (see <code>window.mcmc</code>)
<code>digits</code>	number of digits for output
<code>warn</code>	logical; should warnings be given? Default is TRUE. (Note: this applies only to warnings given directly by JointAI .)
<code>mess</code>	logical; should messages be given? Default is TRUE. (Note: this applies only to messages given directly by JointAI .)
<code>...</code>	Arguments passed on to <code>mcmcse::mcse.mat</code>
<code>size</code>	represents the batch size in “bm” and the truncation point in “bartlett” and “tukey”. Default is NULL which implies that an optimal batch size is calculated using the <code>batchSize()</code> function. Can take character values of “sqrt” and “cuberoot” or any numeric value between 1 and $n/2$. “sqrt” means size is $\text{floor}(n^{1/2})$ and “cuberoot” means size is $\text{floor}(n^{1/3})$.
<code>g</code>	a function such that $E(g(x))$ is the quantity of interest. The default is NULL, which causes the identity function to be used.
<code>method</code>	any of “bm”, “obm”, “bartlett”, “tukey”. “bm” represents batch means estimator, “obm” represents overlapping batch means estimator with, “bartlett” and “tukey” represents the modified-Bartlett window and the Tukey-Hanning windows for spectral variance estimators.
<code>r</code>	the lugsail parameter that converts a lag window into its lugsail equivalent. Larger values of “r” will typically imply less underestimation of “cov”, but higher variability of the estimator. Default is “r = 3” and “r = 1, 2” are good choices. “r > 5” is not recommended. Non-integer values are ok.
<code>data_scale</code>	show the Monte Carlo error of the sample transformed back to the scale of the data (TRUE) or on the sampling scale (this requires the argument <code>keep_scaled_mcmc = TRUE</code> in the JointAI model)

plotpars optional; list of parameters passed to `plot()`
 ablinepars optional; list of parameters passed to `abline()`

Value

An object of class `MCElist` with elements `unscaled`, `scaled` and `digits`. The first two are matrices with columns `est` (posterior mean), `MCSE` (Monte Carlo error), `SD` (posterior standard deviation) and `MCSE/SD` (Monte Carlo error divided by post. standard deviation.)

Methods (by generic)

- `plot`: plot Monte Carlo error

Note

Lesaffre & Lawson (2012) [p. 195] suggest the Monte Carlo error of a parameter should not be more than 5% of the posterior standard deviation of this parameter (i.e., $MCSE/SD \leq 0.05$).

References

Lesaffre, E., & Lawson, A. B. (2012). *Bayesian Biostatistics*. John Wiley & Sons.

See Also

The vignette [Parameter Selection](#) provides some examples how to specify the argument subset.

Examples

```
mod <- lm_imp(y ~ C1 + C2 + M2, data = wideDF, n.iter = 100)
MC_error(mod)
plot(MC_error(mod), ablinepars = list(lty = 2))
```

md_pattern

Missing data pattern

Description

Obtain a plot of the pattern of missing data and/or return the pattern as a matrix.

Usage

```
md_pattern(
  data,
  color = c(grDevices::grey(0.1), grDevices::grey(0.7)),
  border = grDevices::grey(0.5),
  plot = TRUE,
  pattern = FALSE,
  print_xaxis = TRUE,
  ylab = "Number of observations per pattern",
  print_yaxis = TRUE,
  legend.position = "bottom",
  ...
)
```

Arguments

data	data frame
color	vector of length two, that specifies the color used to indicate observed and missing values (in that order)
border	color of the grid
plot	logical; should the missing data pattern be plotted? (default is TRUE)
pattern	logical; should the missing data pattern be returned as matrix? (default is FALSE)
print_xaxis, print_yaxis	logical; should the x-axis (below the plot) and y-axis (on the right) be printed?
ylab	y-axis label
legend.position	the position of legends ("none", "left", "right", "bottom", "top", or two-element numeric vector)
...	optional additional parameters, currently not used

Note

This function requires the **ggplot2** package to be installed.

See Also

See the vignette [Visualizing Incomplete Data](#) for more examples.

Examples

```
op <- par(mar = c(3, 1, 1.5, 1.5), mgp = c(2, 0.6, 0))
md_pattern(wideDF)
par(op)
```

`model_imp`*Joint analysis and imputation of incomplete data*

Description

Functions to estimate (generalized) linear and (generalized) linear mixed models, ordinal and ordinal mixed models, and parametric (Weibull) as well as Cox proportional hazards survival models using MCMC sampling, while imputing missing values.

Usage

```
lm_imp(  
  formula,  
  data,  
  n.chains = 3,  
  n.adapt = 100,  
  n.iter = 0,  
  thin = 1,  
  monitor_params = NULL,  
  auxvars = NULL,  
  refcats = NULL,  
  models = NULL,  
  no_model = NULL,  
  trunc = NULL,  
  ridge = FALSE,  
  ppc = TRUE,  
  seed = NULL,  
  inits = NULL,  
  parallel = FALSE,  
  n.cores = NULL,  
  scale_vars = NULL,  
  scale_pars = NULL,  
  hyperpars = NULL,  
  modelname = NULL,  
  modeldir = NULL,  
  keep_model = FALSE,  
  overwrite = NULL,  
  quiet = TRUE,  
  progress.bar = "text",  
  warn = TRUE,  
  mess = TRUE,  
  keep_scaled_mcmc = FALSE,  
  ...  
)  
  
glm_imp(  
  formula,
```

```
    family,
    data,
    n.chains = 3,
    n.adapt = 100,
    n.iter = 0,
    thin = 1,
    monitor_params = NULL,
    auxvars = NULL,
    refcats = NULL,
    models = NULL,
    no_model = NULL,
    trunc = NULL,
    ridge = FALSE,
    ppc = TRUE,
    seed = NULL,
    inits = NULL,
    parallel = FALSE,
    n.cores = NULL,
    scale_vars = NULL,
    scale_pars = NULL,
    hyperpars = NULL,
    modelname = NULL,
    modeldir = NULL,
    keep_model = FALSE,
    overwrite = NULL,
    quiet = TRUE,
    progress.bar = "text",
    warn = TRUE,
    mess = TRUE,
    keep_scaled_mcmc = FALSE,
    ...
)

c1m_imp(
  fixed,
  data,
  n.chains = 3,
  n.adapt = 100,
  n.iter = 0,
  thin = 1,
  monitor_params = NULL,
  auxvars = NULL,
  refcats = NULL,
  models = NULL,
  no_model = NULL,
  trunc = NULL,
  ridge = FALSE,
  ppc = TRUE,
```

```
seed = NULL,  
inits = NULL,  
parallel = FALSE,  
n.cores = NULL,  
scale_vars = NULL,  
scale_pars = NULL,  
hyperpars = NULL,  
modelname = NULL,  
modeldir = NULL,  
keep_model = FALSE,  
overwrite = NULL,  
quiet = TRUE,  
progress.bar = "text",  
warn = TRUE,  
mess = TRUE,  
keep_scaled_mcmc = FALSE,  
...  
)  
  
lme_imp(  
  fixed,  
  data,  
  random,  
  n.chains = 3,  
  n.adapt = 100,  
  n.iter = 0,  
  thin = 1,  
  monitor_params = NULL,  
  auxvars = NULL,  
  refcats = NULL,  
  models = NULL,  
  no_model = NULL,  
  trunc = NULL,  
  ridge = FALSE,  
  ppc = TRUE,  
  seed = NULL,  
  inits = NULL,  
  parallel = FALSE,  
  n.cores = NULL,  
  scale_vars = NULL,  
  scale_pars = NULL,  
  hyperpars = NULL,  
  modelname = NULL,  
  modeldir = NULL,  
  keep_model = FALSE,  
  overwrite = NULL,  
  quiet = TRUE,  
  progress.bar = "text",
```



```
warn = TRUE,  
mess = TRUE,  
keep_scaled_mcmc = FALSE,  
...  
)
```

```
glme_imp(  
  fixed,  
  data,  
  random,  
  family,  
  n.chains = 3,  
  n.adapt = 100,  
  n.iter = 0,  
  thin = 1,  
  monitor_params = NULL,  
  auxvars = NULL,  
  refcats = NULL,  
  models = NULL,  
  no_model = NULL,  
  trunc = NULL,  
  ridge = FALSE,  
  ppc = TRUE,  
  seed = NULL,  
  inits = NULL,  
  parallel = FALSE,  
  n.cores = NULL,  
  scale_vars = NULL,  
  scale_pars = NULL,  
  hyperpars = NULL,  
  modelname = NULL,  
  modeldir = NULL,  
  keep_model = FALSE,  
  overwrite = NULL,  
  quiet = TRUE,  
  progress.bar = "text",  
  warn = TRUE,  
  mess = TRUE,  
  keep_scaled_mcmc = FALSE,  
  ...  
)
```

```
clmm_imp(  
  fixed,  
  data,  
  random,  
  n.chains = 3,  
  n.adapt = 100,
```

```
n.iter = 0,  
thin = 1,  
monitor_params = NULL,  
auxvars = NULL,  
refcats = NULL,  
models = NULL,  
no_model = NULL,  
trunc = NULL,  
ridge = FALSE,  
ppc = TRUE,  
seed = NULL,  
inits = NULL,  
parallel = FALSE,  
n.cores = NULL,  
scale_vars = NULL,  
scale_pars = NULL,  
hyperpars = NULL,  
modelname = NULL,  
modeldir = NULL,  
keep_model = FALSE,  
overwrite = NULL,  
quiet = TRUE,  
progress.bar = "text",  
warn = TRUE,  
mess = TRUE,  
keep_scaled_mcmc = FALSE,  
...  
)  
  
survreg_imp(  
  formula,  
  data,  
  n.chains = 3,  
  n.adapt = 100,  
  n.iter = 0,  
  thin = 1,  
  monitor_params = NULL,  
  auxvars = NULL,  
  refcats = NULL,  
  models = NULL,  
  no_model = NULL,  
  trunc = NULL,  
  ridge = FALSE,  
  ppc = TRUE,  
  seed = NULL,  
  inits = NULL,  
  parallel = FALSE,  
  n.cores = NULL,
```

```
    scale_vars = NULL,  
    scale_pars = NULL,  
    hyperpars = NULL,  
    modelname = NULL,  
    modeldir = NULL,  
    keep_model = FALSE,  
    overwrite = NULL,  
    quiet = TRUE,  
    progress.bar = "text",  
    warn = TRUE,  
    mess = TRUE,  
    keep_scaled_mcmc = FALSE,  
    ...  
)
```

```
coxph_imp(  
  formula,  
  data,  
  n.chains = 3,  
  n.adapt = 100,  
  n.iter = 0,  
  thin = 1,  
  monitor_params = NULL,  
  auxvars = NULL,  
  refcats = NULL,  
  models = NULL,  
  no_model = NULL,  
  trunc = NULL,  
  ridge = FALSE,  
  ppc = TRUE,  
  seed = NULL,  
  inits = NULL,  
  parallel = FALSE,  
  n.cores = NULL,  
  scale_vars = NULL,  
  scale_pars = NULL,  
  hyperpars = NULL,  
  modelname = NULL,  
  modeldir = NULL,  
  keep_model = FALSE,  
  overwrite = NULL,  
  quiet = TRUE,  
  progress.bar = "text",  
  warn = TRUE,  
  mess = TRUE,  
  keep_scaled_mcmc = FALSE,  
  ...  
)
```

Arguments

formula	a two sided model formula (see formula)
data	a <code>data.frame</code>
n.chains	the number of MCMC chains to be used
n.adapt	the number of iterations for adaptation of the MCMC samplers (see also adapt)
n.iter	the number of iterations of the MCMC chain (after adaptation; see also coda.samples)
thin	thinning interval (see window.mcmc)
monitor_params	named vector specifying which parameters should be monitored (see details)
auxvars	optional one-sided formula of variables that should be used as predictors in the imputation procedure (and will be imputed if necessary) but are not part of the analysis model
refcats	optional; either one of "first", "last", "largest" (which sets the category for all categorical variables) or a named list specifying which category should be used as reference category for each of the categorical variables. Options are the category label, the category number, or one of "first" (the first category), "last" (the last category) or "largest" (chooses the category with the most observations). Default is "first". (See also set_refcat)
models	optional named vector specifying the types of models for (incomplete) covariates. This arguments replaces the argument <code>meth</code> used in earlier versions. If NULL (default) models will be determined automatically based on the class of the respective columns of <code>data</code> .
no_model	names of variables for which no model should be specified. Note that this is only possible for completely observed variables and implies the assumptions of independence between the excluded variable and the incomplete variables.
trunc	optional named list specifying the limits of truncation for the distribution of the named incomplete variables (see the vignette ModelSpecification)
ridge	logical; should the parameters of the main model be penalized using ridge regression? Default is FALSE
ppc	logical: should monitors for posterior predictive checks be set? (not yet used)
seed	optional seed value for reproducibility
inits	optional specification of initial values in the form of a list or a function (see jags.model). If omitted, initial values will be generated automatically by JAGS. It is an error to supply an initial value for an observed node.
parallel	logical; should the chains be sampled using parallel computation? Default is FALSE
n.cores	number of cores to use for parallel computation; if left empty all except two cores will be used
scale_vars	optional; named vector of (continuous) variables that will be scaled (such that mean = 0 and sd = 1) to improve convergence of the MCMC sampling. Default is that all continuous variables that are not transformed by a function (e.g. <code>log()</code> , <code>ns()</code>) will be scaled. Variables for which a log-normal model is used are only scaled with regards to the standard deviation, but not centered. Variables modeled with a Gamma or beta distribution are not scaled. If set to FALSE no scaling will be done.

scale_pars	optional matrix of parameters used for centering and scaling of continuous covariates. If not specified, this will be calculated automatically. If FALSE, no scaling will be done.
hyperpars	list of hyperparameters, as obtained by <code>default_hyperpars()</code> ; only needs to be supplied if hyperparameters other than the default should be used
modelName	optional; character string specifying the name of the model file (including the ending, either .R or .txt). If unspecified a random name will be generated.
modelDir	optional; directory containing the model file or directory in which the model file should be written. If unspecified a temporary directory will be created.
keep_model	logical; whether the created JAGS model should be saved or removed from the disk (FALSE; default) when the sampling has finished.
overwrite	logical; whether an existing model file with the specified <code><modelDir>/<modelName></code> should be overwritten. If set to FALSE and a model already exists, that model will be used. If unspecified (NULL) and a file exists, the user is asked for input on how to proceed.
quiet	if TRUE then messages generated during compilation will be suppressed, as well as the progress bar during adaptation (see <code>jags.model</code>)
progress.bar	character string specifying the type of progress bar. Possible values are "text", "gui", and "none" (see <code>update</code>). Note: when sampling is performed in parallel it is currently not possible to display a progress bar.
warn	logical; should warnings be given? Default is TRUE. (Note: this applies only to warnings given directly by JointAI .)
mess	logical; should messages be given? Default is TRUE. (Note: this applies only to messages given directly by JointAI .)
keep_scaled_mcmc	should the "original" MCMC sample (i.e., the scaled version returned by <code>coda.samples()</code>) be kept? (The MCMC sample that is re-scaled to the scale of the data is always kept.)
...	additional, optional arguments
family	only for <code>glm_imp</code> and <code>glmm_imp</code> : a description of the distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See <code>family</code> and the 'Details' section below.)
fixed	a two sided formula describing the fixed-effects part of the model (see <code>formula</code>)
random	only for multi-level models: a one-sided formula of the form $\sim x_1 + \dots + x_n g$, where $x_1 + \dots + x_n$ specifies the model for the random effects and g the grouping variable

Value

An object of class `JointAI`.

Details

See also the vignettes [Model Specification](#), [MCMC Settings](#) and [Parameter Selection](#).

Implemented distribution families and link functions for `glm_imp()` and `glme_imp()`:

gaussian	with links: identity, log
binomial	with links: logit, probit, log, cloglog
Gamma	with links: inverse, identity, log
poisson	with links: log, identity

Imputation methods: Implemented imputation models that can be chosen in the argument models are:

norm	linear model
lognorm	log-normal model for skewed continuous data
gamma	gamma model (with log-link) for skewed continuous data
beta	beta model (with logit-link) for skewed continuous data in (0, 1)
logit	logistic model for binary data
multilogit	multinomial logit model for unordered categorical variables
cumlogit	cumulative logit model for ordered categorical variables
lmm	linear mixed model for continuous longitudinal covariates
glmm_lognorm	log-normal mixed model for skewed longitudinal covariates
glmm_gamma	Gamma mixed model for skewed longitudinal covariates
glmm_logit	logit mixed model for binary longitudinal covariates
glmm_poisson	Poisson mixed model for longitudinal count covariates
clmm	cumulative logit mixed model for longitudinal ordered factors

When models are specified for only a subset of the incomplete or longitudinal covariates involved in a model, the default choices are used for the unspecified variables.

Parameters to follow (monitor_params): See also the vignette: [Parameter Selection](#)

Named vector specifying which parameters should be monitored. This can be done either directly by specifying the name of the parameter or indirectly by one of the key words selecting a set of parameters. Except for other, in which parameter names are specified directly, parameter (groups) are just set as TRUE or FALSE. If left unspecified, `monitor_params = c("analysis_main" = TRUE)` will be used.

name/key word	what is monitored
analysis_main	betas and sigma_y (and D in multi-level models)
analysis_random	ranef, D, invD, RinVD
imp_pars	alphas, tau_imp, gamma_imp, delta_imp
imps	imputed values
betas	regression coefficients of the analysis model
tau_y	precision of the residuals from the analysis model
sigma_y	standard deviation of the residuals from the analysis model
ranef	random effects b
D	covariance matrix of the random effects
invD	inverse of D
RinVD	matrix in the prior for invD
alphas	regression coefficients in the covariate models
tau_imp	precision parameters of the residuals from covariate models
gamma_imp	intercepts in ordinal covariate models
delta_imp	increments of ordinal intercepts
other	additional parameters

For example:

`monitor_params = c(analysis_main = TRUE, tau_y = TRUE, sigma_y = FALSE)` would monitor the regression parameters `betas` and the residual precision `tau_y` instead of the residual standard deviation `sigma_y`.

`monitor_params = c(imps = TRUE)` would monitor `betas`, `tau_y`, and `sigma_y` (because `analysis_main = TRUE` by default) as well as the imputed values.

Note

Coding of variables:: The default imputation methods are chosen based on the `class` of each of the incomplete variables, distinguishing between `numeric`, `factor` with two levels, `unordered factor` with >2 levels and `ordered factor` with >2 levels.

When a continuous variable has only two different values it is assumed to be binary and its coding and default (imputation) model will be changed accordingly. This behavior can be overwritten specifying a model type via the argument `models`.

Variables of type `logical` are automatically converted to `unordered factors`.

Contrary to base R behavior, dummy coding (i.e., `contr.treatment` contrasts) are used for ordered factors in any linear predictor. It is not possible to overwrite this behavior using the base R contrasts specification. However, since the order of levels in an ordered factor contains information relevant to the imputation of missing values, it is important that incomplete ordinal variables are coded as such.

Non-linear effects and transformation of variables:: **JointAI** handles non-linear effects, transformation of covariates and interactions the following way:

When, for instance, the model formula contains the function `log(x)` and `x` has missing values, `x` will be imputed and used in the linear predictor of models for covariates, i.e., it is assumed that the other variables have a linear association with `x` but not with `log(x)`. The `log()` of the observed and imputed values of `x` is calculated and used in the linear predictor of the analysis model.

If, instead of using `log(x)` in the model formula, a pre-calculated variable `logx` is used instead, this variable is imputed directly and used in the linear predictors of all models, implying that variables that have `logx` in their linear predictors have a linear association with `logx` but not with `x`.

When different transformations of the same incomplete variable are used in one model it is strongly discouraged to calculate these transformations beforehand and supply them as different variables. If, for example, a model formula contains both `x` and `x2` (where `x2 = x^2`), they are treated as separate variables and imputed with separate models. Imputed values of `x2` are thus not equal to the square of imputed values of `x`. Instead, `x` and `I(x^2)` should be used in the model formula. Then only `x` is imputed and used in the linear predictor of models for other incomplete variables, and `x^2` is calculated from the imputed values of `x` internally.

The same applies to interactions involving incomplete variables.

Sequence of covariate models:: The default order is incomplete baseline covariates, complete longitudinal covariates, incomplete longitudinal covariates, and within each group variables are ordered according to the proportion of missing values (increasing).

Not (yet) possible::

- multiple nesting levels of random effects (nested or crossed)
- prediction (using `predict`) conditional on random effects
- the use of splines for incomplete variables
- the use of `pspline`, `frailty`, `cluster` or `strata` in survival models
- left censored or interval censored data

See Also

[set_refcat](#), [get_models](#), [traceplot](#), [densplot](#), [summary.JointAI](#), [MC_error](#), [GR_crit](#), [predict.JointAI](#), [add_samples](#), [JointAIObject](#), [add_samples](#), [parameters](#), [list_models](#)

Vignettes

- [Minimal Example](#)
- [Model Specification](#)
- [Parameter Selection](#)
- [After Fitting](#)

Examples

```
# Example 1: Linear regression with incomplete covariates
mod1 <- lm_imp(y ~ C1 + C2 + M1 + B1, data = wideDF, n.iter = 100)
```

```
# Example 2: Logistic regression with incomplete covariates
mod2 <- glm_imp(B1 ~ C1 + C2 + M1, data = wideDF,
               family = binomial(link = "logit"), n.iter = 100)
```

```
# Example 3: Linear mixed model with incomplete covariates
mod3 <- lme_imp(y ~ C1 + B2 + c1 + time, random = ~ time|id,
               data = longDF, n.iter = 300)
```

NHANES

National Health and Nutrition Examination Survey (NHANES) Data

Description

This data is a small subset of the data collected within the 2011-2012 wave of the NHANES study, a study designed to assess the health and nutritional status of adults and children in the United States, conducted by the [National Center for Health Statistics](#).

Usage

```
data(NHANES)
```

Format

A data frame with 186 rows and 13 variables:

SBP systolic blood pressure

gender male or female

age in years

race race / Hispanic origin (5 categories)

WC waist circumference in cm

alc alcohol consumption (binary: <1 drink per week vs. >= 1 drink per week)

educ educational level (binary: low vs. high)

creat creatinine concentration in mg/dL

albu albumin concentration in g/dL

uricacid uric acid concentration in mg/dL

bili bilirubin concentration in mg/dL

occup occupational status (3 categories)

smoke smoking status (3 ordered categories)

Note

The subset provided here was selected and re-coded to facilitate demonstration of the functionality of the JointAI package, and no clinical conclusions should be derived from it.

Source

National Center for Health Statistics (NCHS) (2011 - 2012). National Health and Nutrition Examination Survey Data. URL <https://www.cdc.gov/nchs/nhanes/>.

Examples

```
summary(NHANES)
```

parameters

Parameter names of an JointAI object

Description

Returns the names of the parameters/nodes of an object of class 'JointAI' for which a monitor is set.

Usage

```
parameters(object, mess = TRUE, warn = TRUE)
```

Arguments

object	object inheriting from class 'JointAI'
mess	logical; should messages be given? Default is TRUE. (Note: this applies only to messages given directly by JointAI .)
warn	logical; should warnings be given? Default is TRUE. (Note: this applies only to warnings given directly by JointAI .)

Examples

```
# (does not need MCMC samples to work, so we will set n.adapt = 0 and
# n.iter = 0 to reduce computational time)
mod1 <- lm_imp(y ~ C1 + C2 + M2 + O2 + B2, data = wideDF, n.adapt = 0, n.iter = 0, mess = FALSE)

parameters(mod1)
```

plot.JointAI *Plot an object object inheriting from class 'JointAI'*

Description

Plot an object object inheriting from class 'JointAI'

Usage

```
## S3 method for class 'JointAI'
plot(x, ...)
```

Arguments

x	object inheriting from class 'JointAI'
...	currently not used

Examples

```
mod <- lm_imp(y ~ C1 + C2 + B1, data = wideDF, n.iter = 100)
plot(mod)
```

plot_all

Visualize the distribution of all variables in the dataset

Description

This function plots a grid of histograms (for continuous variables) and barplots (for categorical variables) and labels it with the proportion of missing values in each variable.

Usage

```
plot_all(
  data,
  nrow = NULL,
  ncol = NULL,
  fill = grDevices::grey(0.8),
  border = "black",
  allNA = FALSE,
  use_level = FALSE,
  idvar,
  xlab = "",
  ylab = "frequency",
  ...
)
```

Arguments

data	a <code>data.frame</code> (or a matrix)
nrow	optional number of rows and columns in the plot layout; automatically chosen if unspecified
ncol	optional number of rows and columns in the plot layout; automatically chosen if unspecified
fill	color the histograms and bars are filled with
border	color of the borders of the histograms and bars
allNA	logical; if FALSE (default) the proportion of missing values is only given for variables that have missing values, if TRUE it is given for all variables
use_level	logical; should the multi-level structure be taken into account? This requires specification of the argument <code>idvar</code> .
idvar	name of the column that specifies the multi-level grouping structure
xlab	labels for the x- and y-axis
ylab	labels for the x- and y-axis
...	additional parameters passed to <code>barplot</code> and <code>hist</code>

See Also

Vignette: [Visualizing Incomplete Data](#)

Examples

```
op <- par(mar = c(2,2,3,1), mgp = c(2, 0.6, 0))
plot_all(wideDF)
par(op)
```

plot_imp_distr	<i>Plot the distribution of observed and imputed values</i>
----------------	---

Description

Plots densities and barplots of the observed and imputed values in a long-format dataset (multiple imputed datasets stacked onto each other).

Usage

```
plot_imp_distr(
  data,
  imp = "Imputation_",
  id = ".id",
  rownr = ".rownr",
  ncol = NULL,
  nrow = NULL
)
```

Arguments

data	a data.frame containing multiple imputations and the original incomplete data stacked onto each other
imp	the name of the variable specifying the imputation indicator
id	the name of the variable specifying the subject indicator
rownr	the name of a variable identifying which rows correspond to the same observation in the original (unimputed) data
ncol	optional number of rows and columns in the plot layout; automatically chosen if unspecified
nrow	optional number of rows and columns in the plot layout; automatically chosen if unspecified

Examples

```
mod <- lme_imp(y ~ C1 + c2 + B2 + C2, random = ~ 1 | id, data = longDF,
  n.iter = 200, monitor_params = c(imps = TRUE), mess = FALSE)
impDF <- get_MIdat(mod, m = 5)
plot_imp_distr(impDF, id = "id", ncol = 3)
```

predDF

Create a new dataframe for prediction

Description

Build a data.frame for prediction, where one variable varies and all other variables are set to the reference value (median for continuous variables).

Usage

```
predDF(object, ...)  
  
## S3 method for class 'JointAI'  
predDF(object, var, length = 100, ...)  
  
## S3 method for class 'formula'  
predDF(formula, dat, var, length = 100, ...)
```

Arguments

object	object inheriting from class 'JointAI'
...	optional, additional arguments (currently not used)
var	name of variable that should be varying
length	number of values used in the sequence when var is continuous
formula	a two sided model formula (see formula)
dat	original data

See Also

[predict.JointAI](#), [lme_imp](#), [glm_imp](#), [lm_imp](#)

Examples

```
# fit a JointAI model  
mod <- lm_imp(y ~ C1 + C2 + M2, data = wideDF, n.iter = 100)  
  
# generate a dataframe with varying "C2" and reference values for all other variables in the model  
newDF <- predDF(mod, var = "C2")  
  
head(newDF)
```

predict.JointAI *Predict values from an object of class JointAI*

Description

Obtains predictions and corresponding credible intervals from an object of class 'JointAI'.

Usage

```
## S3 method for class 'JointAI'
predict(
  object,
  newdata,
  quantiles = c(0.025, 0.975),
  type = c("link", "response", "prob", "class", "lp", "risk"),
  start = NULL,
  end = NULL,
  thin = NULL,
  exclude_chains = NULL,
  mess = TRUE,
  ...
)
```

Arguments

object	object inheriting from class 'JointAI'
newdata	optional new dataset for prediction. If left empty, the original data is used.
quantiles	quantiles of the predicted distribution of the outcome
type	the type of prediction. The default is on the scale of the linear predictor ("link" or "lp"). For generalized linear (mixed) models type = "response" transforms the predicted values to the scale of the response. For ordinal (mixed) models type may be "prob" (to obtain probabilities per class) or "class" to obtain the class with the highest posterior probability.
start	the first iteration of interest (see window.mcmc)
end	the last iteration of interest (see window.mcmc)
thin	thinning interval (see window.mcmc)
exclude_chains	optional vector of the index numbers of chains that should be excluded
mess	logical; should messages be given? Default is TRUE. (Note: this applies only to messages given directly by JointAI .)
...	currently not used

Details

A model matrix X is created from the model formula (currently fixed effects only) and `newdata`. $X\beta$ is then calculated for each iteration of the MCMC sample in object, i.e., $X\beta$ has `n.iter` rows and `nrow(newdata)` columns. A subset of the MCMC sample can be selected using `start`, `end` and `thin`.

Value

A list with entries `dat`, `fit` and `quantiles`, where `fit` contains the predicted values (mean over the values calculated from the iterations of the MCMC sample), `quantiles` contain the specified quantiles (by default 2.5% and 97.5%), and `dat` is `newdata`, extended with `fit` and `quantiles` (unless prediction for an ordinal outcome is done with `type = "prob"`, in which case the quantiles are an array with three dimensions and are therefore not included in `dat`).

Note

- So far, `predict` cannot calculate predicted values for cases with missing values in covariates. Predicted values for such cases are NA.
- For repeated measures models prediction currently only uses fixed effects.

Functionality will be extended in the future.

See Also

[predDF.JointAI, *_imp](#)

Examples

```
# fit model
mod <- lm_imp(y ~ C1 + C2 + I(C2^2), data = wideDF, n.iter = 100)

# calculate the fitted values
fit <- predict(mod)

# create dataset for prediction
newDF <- predDF(mod, var = "C2")

# obtain predicted values
pred <- predict(mod, newdata = newDF)

# plot predicted values and 95% confidence band
plot(newDF$C2, pred$fit, type = "l", ylim = range(pred$quantiles),
      xlab = "C2", ylab = "predicted values")
matplot(newDF$C2, pred$quantiles, lty = 2, add = TRUE, type = "l", col = 1)
```

residuals.JointAI	<i>Extract residuals from an object of class JointAI</i>
-------------------	--

Description

Extract residuals from an object of class JointAI

Usage

```
## S3 method for class 'JointAI'
residuals(object, type = c("deviance", "response", "working"), ...)
```

Arguments

object	object inheriting from class 'JointAI'
type	type of residuals: "deviance", "response", "working"
...	currently not used

Note

- For mixed models residuals are currently calculated using the fixed effects only.
- For ordinal (mixed) models and parametric survival models only type = "response" is available.
- For Cox proportional hazards models residuals are not yet implemented.

Examples

```
mod <- glm_imp(B1 ~ C1 + C2 + O1, data = wideDF, n.iter = 100,
              family = binomial(), mess = FALSE)
summary(residuals(mod, type = 'response'))
summary(residuals(mod, type = 'working'))
```

set_refcat	<i>Specify reference categories for all categorical covariates in the model</i>
------------	---

Description

The function is a helper function that asks questions and, depending on the answers given by the user, returns the input for the argument refcats in the main analysis functions `*_imp`.

Usage

```
set_refcat(data, formula, covars, auxvars = NULL)
```

Arguments

data	a data.frame
formula	optional; model formula (used to select subset of relevant columns of data)
covars	optional; vector containing the names of relevant columns of data
auxvars	optional; formula containing the names of relevant columns of data that should be considered additionally to the columns occurring in the formula

Examples

```
## Not run:
# Example 1: set reference categories for the whole dataset and choose answer option 3:
set_refcat(data = NHANES)
3

# insert the returned string as argument refcats
mod1 <- lm_imp(SBP ~ age + race + creat + educ, data = NHANES, refcats = 'largest')

# Example 2:
# specify a model formula
fmla <- SBP ~ age + gender + race + bili + smoke + alc

# write the output of set_refcat to an object
ref_mod2 <- set_refcat(data = NHANES, formula = fmla)
4
2
5
1
1

# enter the output in the model specification
mod2 <- lm_imp(formula = fmla, data = NHANES, refcats = ref_mod2, n.adapt = 0)

## End(Not run)
```

 sharedParams

Parameters used by several functions in JointAI.

Description

Parameters used by several functions in JointAI.

Arguments

object	object inheriting from class 'JointAI'
no_model	names of variables for which no model should be specified. Note that this is only possible for completely observed variables and implies the assumptions of independence between the excluded variable and the incomplete variables.

subset	subset of parameters/variables/nodes (columns in the MCMC sample). Uses the same logic as the argument <code>monitor_params</code> in <code>*_imp</code> .
exclude_chains	optional vector of the index numbers of chains that should be excluded
start	the first iteration of interest (see window.mcmc)
end	the last iteration of interest (see window.mcmc)
n.adapt	the number of iterations for adaptation of the MCMC samplers (see also adapt)
n.iter	the number of iterations of the MCMC chain (after adaptation; see also coda.samples)
n.chains	the number of MCMC chains to be used
quiet	if TRUE then messages generated during compilation will be suppressed, as well as the progress bar during adaptation (see jags.model)
thin	thinning interval (see window.mcmc)
nrow, ncol	optional number of rows and columns in the plot layout; automatically chosen if unspecified
use_ggplot	logical; Should ggplot be used instead of the base graphics?
warn	logical; should warnings be given? Default is TRUE. (Note: this applies only to warnings given directly by JointAI .)
mess	logical; should messages be given? Default is TRUE. (Note: this applies only to messages given directly by JointAI .)
xlab, ylab	labels for the x- and y-axis
use_level	logical; should the multi-level structure be taken into account? This requires specification of the argument <code>idvar</code> .
idvar	name of the column that specifies the multi-level grouping structure
keep_aux	logical; Should constant effects of auxiliary variables be kept in the output?
ridge	logical; should the parameters of the main model be penalized using ridge regression? Default is FALSE
parallel	logical; should the chains be sampled using parallel computation? Default is FALSE
n.cores	number of cores to use for parallel computation; if left empty all except two cores will be used
seed	optional seed value for reproducibility
ppc	logical: should monitors for posterior predictive checks be set? (not yet used)

 simLong

Simulated Longitudinal Data in Long and Wide Format

Description

This data was simulated to mimic data from a longitudinal cohort study following mothers and their child from birth until approximately 4 years of age. It contains 2400 observations of 200 mother-child pairs. Children's BMI and head circumference was measured repeatedly and their age in months was recorded at each measurement. Furthermore, the data contain several baseline variables with information on the mothers' demographics and socioeconomic status.

Usage

simLong

simWide

Format

simLong: A data frame in long format with 2400 rows and 16 variables

simWide: A data frame in wide format with 200 rows and 81 variables

Baseline covariates

(in simLong and simWide)

GESTBIR gestational age at birth (in weeks)

ETHN ethnicity (binary: European vs. other)

AGE_M age of the mother at intake

HEIGHT_M height of the mother (in cm)

PARITY number of times the mother has given birth (binary: 0 vs. >=1)

SMOKE smoking status of the mother during pregnancy (3 ordered categories: never smoked during pregnancy, smoked until pregnancy was known, continued smoking in pregnancy)

EDUC educational level of the mother (3 ordered categories: low, mid, high)

MARITAL marital status (3 categories)

ID subject identifier

Long-format variables

(only in simLong)

time measurement occasion/visit (by design, children should be measured at/around 1, 2, 3, 4, 7, 11, 15, 20, 26, 32, 40 and 50 months of age)

age child age at measurement time in months

bmi child BMI

hc child head circumference in cm

hgt child height in cm

wgt child weight in gram

sleep sleeping behavior of the child (3 ordered categories)

Wide-format variables

(only in simWide)

age1, age2, age3, age4, age7, age11, age15, age20, age26, age32, age40, age50 child age at the repeated measurements in months

bmi1, bmi2, bmi3, bmi4, bmi7, bmi11, bmi15, bmi20, bmi26, bmi32, bmi40, bmi50 repeated measurements of child BMI

hc1, hc2, hc3, hc4, hc7, hc11, hc15, hc20, hc26, hc32, hc40, hc50 repeated measurements of child head circumference in cm

hgt1, hgt2, hgt3, hgt4, hgt7, hgt11, hgt15, hgt20, hgt26, hgt32, hgt40, hgt50 repeated measurements of child height in cm

wgt1, wgt2, wgt3, wgt4, wgt7, wgt11, wgt15, wgt20, wgt26, wgt32, wgt40, wgt50 repeated measurements of child weight in gram

sleep1, sleep2, sleep3, sleep4, sleep7, sleep11, sleep15, sleep20, sleep26, sleep32, sleep40, sleep50 repeated measurements of child sleep behavior (3 ordered categories)

Examples

```
summary(simLong)
summary(simWide)
```

```
summary.JointAI
```

```
Summary of an object of class JointAI
```

Description

Obtain and print the summary, (fixed effects) coefficients (coef) and credible interval (confint) for an object of class 'JointAI'.

Usage

```
## S3 method for class 'JointAI'
summary(
  object,
  start = NULL,
  end = NULL,
  thin = NULL,
  quantiles = c(0.025, 0.975),
  subset = NULL,
  exclude_chains = NULL,
  warn = TRUE,
  mess = TRUE,
  ...
)

## S3 method for class 'summary.JointAI'
print(x, digits = max(3, .Options$digits - 4), ...)

## S3 method for class 'JointAI'
coef(
```

```

    object,
    start = NULL,
    end = NULL,
    thin = NULL,
    subset = NULL,
    exclude_chains = NULL,
    warn = TRUE,
    mess = TRUE,
    ...
)

## S3 method for class 'JointAI'
confint(
  object,
  parm = NULL,
  level = 0.95,
  quantiles = NULL,
  start = NULL,
  end = NULL,
  thin = NULL,
  subset = NULL,
  exclude_chains = NULL,
  warn = TRUE,
  mess = TRUE,
  ...
)

## S3 method for class 'JointAI'
print(x, digits = max(4, getOption("digits") - 4), ...)

```

Arguments

object	object inheriting from class 'JointAI'
start	the first iteration of interest (see window.mcmc)
end	the last iteration of interest (see window.mcmc)
thin	thinning interval (see window.mcmc)
quantiles	posterior quantiles
subset	subset of parameters/variables/nodes (columns in the MCMC sample). Uses the same logic as the argument <code>monitor_params</code> in <code>*_imp</code> .
exclude_chains	optional vector of the index numbers of chains that should be excluded
warn	logical; should warnings be given? Default is TRUE. (Note: this applies only to warnings given directly by JointAI .)
mess	logical; should messages be given? Default is TRUE. (Note: this applies only to messages given directly by JointAI .)
...	currently not used
x	an object of class <code>summary.JointAI</code> or <code>JointAI</code>

digits minimal number of *significant* digits, see [print.default](#).
 parm same as subset
 level confidence level (default is 0.95)

See Also

The model fitting functions [lm_imp](#), [glm_imp](#), [clm_imp](#), [lme_imp](#), [glme_imp](#), [survreg_imp](#) and [coxph_imp](#), and the vignette [Parameter Selection](#) for examples how to specify the parameter subset.

Examples

```

mod1 <- lm_imp(y ~ C1 + C2 + M2, data = wideDF, n.iter = 100)

summary(mod1)
coef(mod1)
confint(mod1)

```

traceplot	<i>Traceplot of a JointAI model</i>
-----------	-------------------------------------

Description

Creates a set of traceplots from the MCMC sample of an object of class "JointAI".

Usage

```

traceplot(object, ...)

## S3 method for class 'mcmc.list'
traceplot(object, start = NULL, end = NULL, thin = NULL, ...)

## S3 method for class 'JointAI'
traceplot(
  object,
  start = NULL,
  end = NULL,
  thin = NULL,
  subset = c(analysis_main = TRUE),
  exclude_chains = NULL,
  nrow = NULL,
  ncol = NULL,
  keep_aux = FALSE,
  use_ggplot = FALSE,
  warn = TRUE,
  mess = TRUE,
  ...
)

```

Arguments

object	object inheriting from class 'JointAI'
...	Arguments passed on to <code>graphics::matplot</code>
lty	vector of line types, widths, and end styles. The first element is for the first column, the second element for the second column, etc., even if lines are not plotted for all columns. Line types will be used cyclically until all plots are drawn.
lwd	vector of line types, widths, and end styles. The first element is for the first column, the second element for the second column, etc., even if lines are not plotted for all columns. Line types will be used cyclically until all plots are drawn.
lend	vector of line types, widths, and end styles. The first element is for the first column, the second element for the second column, etc., even if lines are not plotted for all columns. Line types will be used cyclically until all plots are drawn.
col	vector of colors. Colors are used cyclically.
cex	vector of character expansion sizes, used cyclically. This works as a multiple of <code>par("cex")</code> . NULL is equivalent to 1.0.
bg	vector of background (fill) colors for the open plot symbols given by <code>pch = 21:25</code> as in <code>points</code> . The default NA corresponds to the one of the underlying function <code>plot.xy</code> .
xlim	ranges of x and y axes, as in <code>plot</code> .
ylim	ranges of x and y axes, as in <code>plot</code> .
add	logical. If TRUE, plots are added to current one, using <code>points</code> and <code>lines</code> .
verbose	logical. If TRUE, write one line of what is done.
start	the first iteration of interest (see <code>window.mcmc</code>)
end	the last iteration of interest (see <code>window.mcmc</code>)
thin	thinning interval (see <code>window.mcmc</code>)
subset	subset of parameters/variables/nodes (columns in the MCMC sample). Uses the same logic as the argument <code>monitor_params</code> in <code>*_imp</code> .
exclude_chains	optional vector of the index numbers of chains that should be excluded
nrow	optional number of rows and columns in the plot layout; automatically chosen if unspecified
ncol	optional number of rows and columns in the plot layout; automatically chosen if unspecified
keep_aux	logical; Should constant effects of auxiliary variables be kept in the output?
use_ggplot	logical; Should ggplot be used instead of the base graphics?
warn	logical; should warnings be given? Default is TRUE. (Note: this applies only to warnings given directly by JointAI .)
mess	logical; should messages be given? Default is TRUE. (Note: this applies only to messages given directly by JointAI .)

See Also

[summary.JointAI](#), [lme_imp](#), [glm_imp](#), [lm_imp](#), [densplot](#) The vignette [Parameter Selection](#) contains some examples how to specify the parameter subset.

Examples

```
# fit a JointAI model
mod <- lm_imp(y ~ C1 + C2 + M1, data = wideDF, n.iter = 100)

# Example 1: simple traceplot
traceplot(mod)

# Example 2: ggplot version of traceplot
traceplot(mod, use_ggplot = TRUE)

# Example 5: changing how the ggplot version looks (using standard ggplot syntax)
library(ggplot2)

traceplot(mod, use_ggplot = TRUE) +
  theme(legend.position = 'botto') +
  xlab('iteration') +
  ylab('value') +
  scale_color_discrete(name = 'chain')
```

wideDF

Cross-sectional example dataset

Description

A simulated cross-sectional dataset.

Usage

```
data(wideDF)
```

Format

A simulated data frame with 100 rows and 13 variables:

- C1** continuous, complete variable
- C2** continuous, incomplete variable
- B1** binary, complete variable
- B2** binary, incomplete variable

M1 unordered factor; complete variable
M2 unordered factor; incomplete variable
O1 ordered factor; complete variable
O2 ordered factor; incomplete variable
L1 continuous, complete variable
L2 continuous incomplete variable
id id (grouping) variable
time continuous complete variable
y continuous, complete variable

Index

*Topic **datasets**

- longDF, 17
- NHANES, 33
- simLong, 43
- wideDF, 49
- *_imp, 3, 6, 9, 11, 14, 19, 40, 41, 43, 46, 48

- abline, 7, 20
- adapt, 28, 43
- add_samples, 2, 33

- barplot, 36

- clm, 13
- clm_imp, 12, 47
- clm_imp(model_imp), 22
- clmm2, 13
- clmm_imp, 12
- clmm_imp(model_imp), 22
- cluster, 33
- coda.samples, 3, 28, 43
- coef(), 13
- coef.JointAI(summary.JointAI), 45
- confint(), 13
- confint.JointAI(summary.JointAI), 45
- coxph, 13
- coxph_imp, 13, 47
- coxph_imp(model_imp), 22

- default_hyperpars, 4, 13, 29
- densplot, 6, 13, 14, 33, 49
- densplot(), 13

- family, 29
- formula, 10, 28, 29, 38
- frailty, 33

- gelman.diag, 11
- get_MIdat, 8, 14
- get_MIdat(), 13
- get_models, 9, 13, 33

- glm, 13
- glm_imp, 12, 13, 38, 47, 49
- glm_imp(model_imp), 22
- glme_imp, 12, 47
- glme_imp(model_imp), 22
- glmer_imp(model_imp), 22
- GR_crit, 11, 13, 14, 33
- graphics::matplot, 48

- hist, 36

- jags.model, 28, 29, 43
- JointAI, 12, 29
- JointAIObject, 14, 33

- lines, 48
- list_models, 13, 16, 33
- lm, 13
- lm_imp, 12, 13, 38, 47, 49
- lm_imp(model_imp), 22
- lme, 13
- lme_imp, 12, 13, 38, 47, 49
- lme_imp(model_imp), 22
- longDF, 17

- MC_error, 13, 14, 18, 33
- mcmcse::mcse.mat, 19
- md_pattern, 13, 20
- model_imp, 22

- NHANES, 33

- par, 48
- parameters, 13, 33, 34
- plot, 6, 20, 48
- plot.JointAI, 35
- plot.MCElist(MC_error), 18
- plot.xy, 48
- plot_all, 13, 36
- plot_imp_distr, 9, 37
- plot_imp_distr(), 13

points, 48
predDF, 14, 38
predDF.JointAI, 40
predict, 14
predict(), 13
predict.JointAI, 33, 38, 39
print.default, 47
print.JointAI (summary.JointAI), 45
print.summary.JointAI
 (summary.JointAI), 45
pspline, 33

residuals.JointAI, 41

set_refcat, 13, 28, 33, 41
sharedParams, 42
simLong, 43
simWide (simLong), 43
strata, 33
summary, 13, 14
summary(), 13
summary.JointAI, 13, 33, 45, 49
survreg, 13
survreg_imp, 13, 47
survreg_imp (model_imp), 22

traceplot, 13, 14, 33, 47
traceplot(), 13

update, 3, 29

wideDF, 49
window.mcmc, 3, 6, 8, 11, 12, 19, 28, 39, 43,
 46, 48