Package 'JointAI'

March 22, 2018

Version 0.1.0

Title Joint Analysis and Imputation of Incomplete Data

Author Nicole S. Erler

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

Description Provides joint analysis and imputation of linear regression models, generalized linear regression models or linear mixed 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.

License GPL (>= 2)

Date 2018-03-21

BugReports https://github.com/nerler/JointAI

LazyData TRUE

RoxygenNote 6.0.1

Depends rjags (>= 4-6)

Imports MASS, mcmcse, coda

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

Suggests knitr, rmarkdown, mice, foreign

NeedsCompilation no

Repository CRAN

Date/Publication 2018-03-22 09:53:45 UTC

R topics documented:

`add_samples` *Add samples to an object of class JointAI*

Description

Allows to continue sampling from an existing object of class JointAI

Usage

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

Arguments

`object`	object inheriting from class `JointAI`
`n.iter`	number of iterations to monitor
`add`	logical; should the new MCMC samples be added to the existing samples or replace them? If samples are added, `thin` and `var.names` are ignored
`thin`	thinning interval for monitors
`monitor_params`	named vector specifying which parameters should be monitored, see details.
`progress.bar`	type of progress bar. Possible values are `"text"`, `"gui"`, and `"none"`. See Details.

Examples

```
mod <- lm_imp(y~C1 + C2 + M2, data = wideDF, n.iter = 100)
mod_add <- add_samples(mod, n.iter = 200, add = TRUE)

# or to additionally sample imputed values
imps <- add_samples(mod, n.iter = 200, monitor_params = c("imps" = TRUE),
                    add = FALSE)
```

`default_hyperpars` *Get default values for hyperparameters Prints the list of default values for the hyperparameters*

Description

Get default values for hyperparameters Prints the list of default values for the hyperparameters

Usage

```
default_hyperpars(family = "gaussian", link = "identity", nranef = NULL)
```

Arguments

`family`	distribution family of the analysis model (`gaussian`, `binomial`, `poisson` or `Gamma`)
`link`	link function (if the link is already given in the family, e.g. `family = binomial("logit")`) this argument does not need to be specified
`nranef`	number of random effects

Value

A list containing the default hyperparameters for JointAI models. The elements of the list are

analysis_model: hyperparameters for the analysis model

`mu_reg_main`	mean in the priors for regression coefficients
`tau_reg_main`	precision in the priors for regression coefficients
`a_tau_main`	scale parameter in gamma prior for precision of outcome
`b_tau_main`	rate parameter in gamma prior for precision of outcome

Z: hyperparameters for the random effects in mixed models

`RinvD`	scale matrix in Wishart prior (*) for random effects covariance matrix
`KinvD`	degrees of freedom in Wishart prior for random effects covariance matrix
`a_diag_RinvD`	scale parameter in gamma prior for the diagonal elements of `RinvD`

b_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

norm: hyperparameters for normal and lognormal imputation models

mu_reg_norm	mean in the priors for regression coefficients
tau_reg_norm	precision in the priors for regression coefficients
a_tau_norm	scale parameter in gamma prior for precision of imputed variable
b_tau_norm	rate parameter in gamma prior for precision of imputed variable

logit: hyperparameters for logistic imputation 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 imputation 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 imputation 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

---

densplot *Plot posterior density from JointAI model*

Description

Plots a set of densities (per MC chain and coefficient) from the MCMC sample of an object of class JointAI

Usage

```
densplot(object, ...)

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

```
densplot(object, start = NULL, end = NULL, thin = NULL,
  subset = "main", vlines = NULL, nrow = NULL, ncol = NULL, ...)
```

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 monitored parameters (columns in the MCMC sample). Can be specified as a numeric vector of columns, a vector of column names, as subset = "main" or NULL. If NULL, all monitored nodes will be plotted. subset = "main" (default) the main parameters of the analysis model will be plotted (regression coefficients/fixed effects, and, if available, standard deviation of the residual and random effects covariance matrix).
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 v = <x location>, where <x location> is a vector of the same length as the number of plots (see examples).
nrow	optional; number of rows in the plotting layout (determined automatically if not specified)
ncol	optional; number of columns in the plotting layout (determined automatically if not specified)

Examples

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

# densplot without vertical lines
densplot(mod)

# 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))))

# 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)))
```

get_imp_meth *Find default imputation methods and order*

Description

Find default imputation methods and order

Usage

```
get_imp_meth(fixed, random = NULL, data, auxvars = NULL)
```

Arguments

fixed	a two sided (fixed effects) model formula (see formula).
random	only for lme_imp: a one-sided formula of the form ~x1 + ... + xn | g, where x1 + ... + xn specifies the model for the random effects and g the grouping variable
data	a data frame
auxvars	optional vector of variable names that should be used as predictors in the imputation procedure (and will be imputed if necessary) but are not part of the analysis model

Value

a named vector containing those variables in data that have missing values and their assigned default imputation methods, sorted by proportion of missing values

Examples

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

get_MIdat *Extract multiple imputed datasets (and export to SPSS)*

Description

Extracts a dataset containing multiple imputed datasets. 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, start = NULL, seed = NULL, resdir = NULL,
  filename = NULL, export_to_SPSS = FALSE)
```

Arguments

object	object inheriting from class JointAI
m	number of imputed datasets
start	the first iteration of interest (see window.mcmc)
seed	optional seed
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)
export_to_SPSS	logical

Value

A dataframe containing the imputed values (and original data) stacked. The variable Imputation_ identifies the imputations.

Examples

```
mod <- lm_imp(y~C1 + C2 + M2, data = wideDF, n.iter = 100)
MIs <- get_MIdat(mod, m = 3, seed = 123)

## Not run:
# or 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)
```

GR_crit *Gelman-Rubin criterion for convergence*

Description

Gelman-Rubin criterion for convergence (uses gelman.diag)

Usage

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

Arguments

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 monitored parameters (columns in the MCMC sample). Can be specified as a numeric vector of columns, a vector of column names, as subset = "main" or NULL. If NULL, all monitored nodes will be plotted. subset = "main" (default) the main parameters of the analysis model will be plotted (regression coefficients/fixed effects, and, if available, standard deviation of the residual and random effects covariance matrix).
start	the first iteration of interest (see window.mcmc)
end	the last iteration of interest (see window.mcmc)
thin	thinning interval (see window.mcmc)
...	currently not used

References

Gelman, A., Meng, X. L., & Stern, H. (1996). Posterior predictive assessment of model fitness via realized discrepancies. *Statistica Sinica*, 733-760.

Examples

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

JointAI *JointAI: Joint Analysis and Imputation of Missing Values*

Description

The JointAI package performs simultaneous imputation and inference for incomplete data using the Bayesian framework. Distributions for incomplete variables are specified automatically and modeled jointly with the analysis model.

Main functions

The package has three main functions, `lm_imp`, `glm_imp` and `lme_imp`, that allow analysis using linear regression, generalized linear regression and linear mixed effects models. As far as possible, the specification of the functions is the same as the specification of their complete data versions (`lm`, `glm` and `lme`).

Results can be summarized and printed with `summary.JointAI`, and visualized using `traceplot` or `densplot`.

Evaluation and export

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

Imputed data can be exported to SPSS data using `get_MIdat`.

References

Erler, N. S., Rizopoulos, D., Rosmalen, J. V., Jaddoe, V. W., Franco, O. H., & Lesaffre, E. M. (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.

`JointAIObject` *Fitted object of class JointAI*

Description

An object returned by one of the functions `lm_imp()`, `glm_imp()` or `lme_imp()`.

Value

`analysis_type`	`lm`, `glm` or `lme`, with attributes `family` and `link`
`data`	the original dataset
`meth`	named vector specifying imputation methods and sequence
`fixed`	supplied fixed effects structure
`random`	supplied random effects structure
`Mlist`	a list of matrices that contain the data split up into outcome (y), cross-sectional main effects (Xc), cross-sectional interactions (Xic), longitudinal main effects (Xl), longitudinal interactions (Xil), categorical incomplete variables (Xcat), transformed cross-sectional variables (Xtrafo), random effects design matrix (Z), the vector of variables to be scaled (scale_vars), reference values and dummies for categorical variables (refs), specification for transformations (trafos), specification for hierarchical centering (hc_list), vector of auxiliary variables (auxvars), grouping specification (groups), updated fixed effects structure (fixed2), names of updated design matrix (X2_names)
`refcats`	A list naming the reference categories for all categorical covariates.

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
	MCMCpackage which package has been used (at the moment only JAGS is implemented)
	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 used
data_list	list with data that was passed to JAGS
scale_pars	matrix with parameters used to center and scale the continuous variables
hyperpars	a list containing the values of the hyperparameters used
model	JAGS model
sample	MCMC sample (Note: if continuous variables have been scaled during the sampling, the posterior sample here is on the scaled scale, not on the original scale.)
MCMC	if scaling was done: MCMC sample, scaled back to original scale
time	the computational time used for the sampling (adaptive phase + sampling)
call	the original call

---

longDF *Longitudinal example dataset*

Description

Longitudinal example dataset

Usage

```
data(longDF)
```

Format

A simulated data frame with 318 rows and 13 variables:

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

L1 continuous, complete longitudinal variable

L2 continuous incomplete longitudinal variable

id id (grouping) variable

time continuous complete longitudinal variable

y continuous, longitudinal (outcome) variable

`MC_error` *Monte Carlo error*

Description

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

Usage

```
MC_error(x, subset = "main", start = NULL, end = NULL, thin = NULL,
  digits = 2, ...)

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

Arguments

`x`	object inheriting from class `JointAI`
`subset`	subset of monitored parameters (columns in the MCMC sample). Can be specified as a numeric vector of columns, a vector of column names, as `subset = "main"` or `NULL`. If `NULL`, all monitored nodes will be plotted. `subset = "main"` (default) the main parameters of the analysis model will be plotted (regression coefficients/fixed effects, and, if available, standard deviation of the residual and random effects covariance matrix).
`start`	the first iteration of interest (see `window.mcmc`)
`end`	the last iteration of interest (see `window.mcmc`)
`thin`	thinning interval (see `window.mcmc`)
`digits`	number of digits for output
`...`	Arguments passed on to `mcmcse::mcse.mat`

size the batch size. The default value is "sqroot", which uses the square root of the sample size. "cuberoot" will cause the function to use the cube root of the sample size. A numeric value may be provided if neither "sqroot" nor "cuberoot" is satisfactory.

g a function such that $E(g(x))$ is the quantity of interest. The default is NULL, which causes the identity function to be used.

method the method used to compute the standard error. This is one of "bm" (batch means, the default), "obm" (overlapping batch means), "tukey" (spectral variance method with a Tukey-Hanning window), or "bartlett" (spectral variance method with a Bartlett window).

`scaled` use the scaled or unscaled version, default is TRUE

`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.

Examples

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

md_pattern *Missing data pattern*

Description

Plot the pattern of missing data. The missing data pattern is determined using the function `md.pattern` from the `mice` package.

Usage

```
md_pattern(data, plot = TRUE, xlab = "", ylab = "", xaxis_pars = list(),
  yaxis_pars = list(), printN = TRUE, print = TRUE, ...)
```

Arguments

`data`	data frame
`plot`	logical; should the missing data pattern be plotted?
`xlab`	label for the x-axis
`ylab`	label for the y-axis
`xaxis_pars`	list of optional parameters for the x-axis
`yaxis_pars`	list of optional parameters for the y-axis
`printN`	logical; should the title "N" of the y-axis be printed?
`print`	should the missing data pattern be returned as a matrix?
`...`	optional additional parameters passed to `image()`

Examples

```
md_pattern(wideDF)

par(mar = c(3, 1, 1.5, 1.5), mgp = c(2, 0.6, 0))
md_pattern(longDF, yaxis_pars = list(cex.axis = 0.8))
```

model_imp *Joint analysis and imputation of incomplete data*

Description

`lm_imp`, `glm_imp` and `lme_imp` estimate linear, generalized linear and linear mixed models, respectively, using MCMC sampling.

Usage

```
lm_imp(formula, data, n.chains = 3, n.adapt = 100, n.iter = 0, thin = 1,
  monitor_params = NULL, inits = TRUE, modelname = NULL,
  modeldir = NULL, overwrite = FALSE, keep_model = FALSE, quiet = TRUE,
  progress.bar = "text", warn = TRUE, auxvars = NULL, meth = NULL,
  refcats = NULL, scale_vars = NULL, hyperpars = NULL, ...)

glm_imp(formula, family, data, n.chains = 3, n.adapt = 100, n.iter = 0,
  thin = 1, monitor_params = NULL, inits = TRUE, modelname = NULL,
  modeldir = NULL, overwrite = FALSE, keep_model = FALSE, quiet = TRUE,
  progress.bar = "text", warn = TRUE, auxvars = NULL, meth = NULL,
  refcats = NULL, scale_vars = NULL, hyperpars = NULL, ...)

lme_imp(fixed, data, random, n.chains = 3, n.adapt = 100, n.iter = 0,
  thin = 1, monitor_params = NULL, inits = TRUE, modelname = NULL,
  modeldir = NULL, overwrite = FALSE, keep_model = FALSE, quiet = TRUE,
  progress.bar = "text", warn = TRUE, auxvars = NULL, meth = NULL,
  refcats = NULL, scale_vars = NULL, hyperpars = NULL, ...)
```

Arguments

formula	a two sided model formula (see formula)
data	a data frame
n.chains	the number of parallel chains for the model
n.adapt	the number of iterations for adaptation. See adapt for details. If n.adapt = 0 then no adaptation takes place.
n.iter	number of iterations to monitor
thin	thinning interval for monitors
monitor_params	named vector specifying which parameters should be monitored, see details.
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. It is an error to supply an initial value for an observed node.
modelname	optional; character string specifying the name of model file (including the ending, either .R or .txt). If unspecified a random name will be generated.
modeldir	optional; directory containing model file. If unspecified a temporary directory will be created.
overwrite	logical; whether an existing model file with the specified modeldir/modelname should be overwritten. If set to FALSE (default) and a model already exists, that model will be used.
keep_model	logical; whether the created JAGS model should be saved or removed from the disk (FALSE; default) when the sampling has finished.
quiet	if TRUE then messages generated during compilation will be suppressed, as well as the progress bar during adaptation.
progress.bar	type of progress bar. Possible values are "text", "gui", and "none". See Details.

warn	logical; should warnings and messages be suppressed; default is TRUE. Note: this applies only to warnings and messages given directly by **JointAI**.
auxvars	optional vector of variable names that should be used as predictors in the imputation procedure (and will be imputed if necessary) but are not part of the analysis model
meth	optional named vector specifying imputation model types and order. If NULL (default) imputation models will be determined automatically based on the class of the columns of data that contain missing values (see Details). The default order is according to the proportion of missing values (increasing).
refcats	optional; 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, 'first" (the first category) or "largest" (chooses the category with the most observations). Default is "first".
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. log(), ns()) will be scaled. Variables for which "lognorm" is set as imputation model are only scaled with regards to the standard deviation, but not centered. If set to FALSE no scaling will be done.
hyperpars	list of hyperparameters, as obtained by default_hyperpars(); only needs to be supplied if hyperparameters other than the default should be used
...	additional, optional arguments, see below
family	only for glm_imp: 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 family and the 'Details' section below.)
fixed	a two sided formula describing the fixed-effects part of the model (see formula)
random	only for lme_imp: a one-sided formula of the form ~x1 + ... + xn | g, where x1 + ... + xn specifies the model for the random effects and g the grouping variable

Value

An object of class JointAI

Optional arguments

There are some optional parameters that can be passed to ...

scale_pars optional matrix of parameters used for centering and scaling continuous covariates. If not specified, this will be calculated automatically. If FALSE, no scaling will be done.

Details

Implemented distribution families and link functions for glm_imp():

gaussian	with links: identity, log

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

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

norm	linear model
lognorm	log-linear model for skewed continuous data
logit	logistic model for binary data
multinomial	multinomial logit model for unordered categorical variables
ordinal	cumulative logit model for ordered categorical variables

Parameters to follow (monitor_params): 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 summarizing a number 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, tau_y and sigma_y
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
D	covariance matrix of the random effects
invD	inverse of D
RinvD	matrix in prior for invD
alphas	regression coefficients in the imputation models
tau_imp	precision parameters of the residuals from imputation models
gamma_imp	intercepts in ordinal imputation models
delta_imp	increments of ordinal intercepts
other	additional parameters

For example:

monitor_params = c("analysis_main" = TRUE, "tau_y" = FALSE) would monitor the regression parameters betas and residual standard deviation sigma_y, but not the residual precision.

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.

See Also

traceplot, densplot, summary.JointAI, MC_error, GR_crit, jags.model, coda.samples, predict.JointAI

Examples

```
mod1 <- lm_imp(y~C1 + C2 + M2, data = wideDF, n.iter = 100)
mod2 <- glm_imp(B1 ~ C1 + C2 + M2, data = wideDF,
                family = binomial(link = "logit"), n.iter = 100)
mod3 <- lme_imp(y ~ C1 + B2 + L1 + time, random = ~ time|id,
                data = longDF, n.iter = 500)
```

---

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(...)

## S3 method for class 'formula'
predDF(formula, dat, var, ...)

## S3 method for class 'JointAI'
predDF(object, var, ...)
```

Arguments

...	optional, additional arguments (currently not used)
formula	model formula (only fixed effects)
dat	original data
var	name of variable that should be varying
object	object inheriting from class JointAI

See Also

predict.JointAI, lme_imp, glm_imp, lm_imp

Examples

```
mod <- lm_imp(y~C1 + C2 + M2, data = wideDF, n.iter = 100)
newDF <- predDF(mod, var = "C2")
```

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

Description

Calculates the expected outcome value for a given set of covariate values and an object of class JointAI, and corresponding 2.5% and 97.5% (or other quantiles) credible intervals.

Usage

```
## S3 method for class 'JointAI'
predict(object, newdata, quantiles = c(0.025, 0.975),
  start = NULL, end = NULL, thin = NULL, ...)
```

Arguments

`object`	object inheriting from class `JointAI`
`newdata`	new dataset for prediction
`quantiles`	quantiles of the predicted distribution of the outcome
`start`	the first iteration of interest (see `window.mcmc`)
`end`	the last iteration of interest (see `window.mcmc`)
`thin`	thinning interval (see `window.mcmc`)
`...`	currently not used

Details

A `model.matrix` X is created from the model formula (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 "fit" and "quantiles", where "fit" contains the column means of $X\beta$ (see details) and "quantiles" contain the specified quantiles (by default 2.5 and 97.5

See Also

`predDF.JointAI`, `lme_imp`, `glm_imp`, `lm_imp`

Examples

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

# 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, t(pred$quantiles), lty = 2, add = TRUE, type = "l", col = 1)
```

---

summary.JointAI *Summary of an object of class JointAI*

Description

Summary of 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 = "main", ...)

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

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 monitored parameters (columns in the MCMC sample). Can be specified as a numeric vector of columns, a vector of column names, as subset = "main" or NULL. If NULL, all monitored nodes will be plotted. subset = "main" (default) the main parameters of the analysis model will be plotted (regression coefficients/fixed effects, and, if available, standard deviation of the residual and random effects covariance matrix).
...	currently not used
x	an object of class summary.JointAI
digits	minimal number of *significant* digits, see print.default.

See Also

The model fitting functions lm_imp, glm_imp, lme_imp

Examples

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

---

traceplot *Traceplot of a JointAI model*

Description

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

Usage

```
traceplot(object, ...)

## S3 method for class 'JointAI'
traceplot(object, start = NULL, end = NULL, thin = NULL,
  subset = "main", nrow = NULL, ncol = NULL, ...)
```

Arguments

object	object inheriting from class JointAI
...	Arguments passed on to graphics::matplot
	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 par("cex"). NULL is equivalent to 1.0.

bg vector of background (fill) colors for the open plot symbols given by `pch = 21:25` as in `points`. The default `NA` corresponds to the one of the underlying function `plot.xy`.

xlim ranges of x and y axes, as in `plot`.

ylim ranges of x and y axes, as in `plot`.

add logical. If `TRUE`, plots are added to current one, using `points` and `lines`.

verbose logical. If `TRUE`, write one line of what is done.

`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 monitored parameters (columns in the MCMC sample). Can be specified as a numeric vector of columns, a vector of column names, as `subset = "main"` or `NULL`. If `NULL`, all monitored nodes will be plotted. `subset = "main"` (default) the main parameters of the analysis model will be plotted (regression coefficients/fixed effects, and, if available, standard deviation of the residual and random effects covariance matrix).
`nrow`	optional; number of rows in the plotting layout (determined automatically if not specified)
`ncol`	optional; number of columns in the plotting layout (determined automatically if not specified)

See Also

`summary.JointAI`, `lme_imp`, `glm_imp`, `lm_imp`

Examples

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

---

wideDF *Cross-sectional example dataset*

Description

Cross-sectional example 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