Package: metapack (via r-universe)

October 21, 2024

Type Package

Title Bayesian Meta-Analysis and Network Meta-Analysis

Version 0.3

Date 2024-01-14

Description Contains functions performing Bayesian inference for meta-analytic and network meta-analytic models through Markov chain Monte Carlo algorithm. Currently, the package implements Hui Yao, Sungduk Kim, Ming-Hui Chen, Joseph G. Ibrahim, Arvind K. Shah, and Jianxin Lin (2015) <doi:10.1080/01621459.2015.1006065> and Hao Li, Daeyoung Lim, Ming-Hui Chen, Joseph G. Ibrahim, Sungduk Kim, Arvind K. Shah, Jianxin Lin (2021) <doi:10.1002/sim.8983>. For maximal computational efficiency, the Markov chain Monte Carlo samplers for each model, written in C++, are fine-tuned. This software has been developed under the auspices of the National Institutes of Health and Merck & Co., Inc., Kenilworth, NJ, USA.

License GPL (>= 3)

Encoding UTF-8

LazyLoad yes

LazyData true

RoxygenNote 7.1.1

NeedsCompilation yes

Imports Rcpp, ggplot2, methods, gridExtra, Formula

Depends R (>= 3.4)

LinkingTo Rcpp, RcppArmadillo, RcppProgress, BH

Suggests knitr, rmarkdown

VignetteBuilder knitr

URL `https://events.stat.uconn.edu/metapack/`

BugReports `https://github.com/daeyounglim/metapack/issues`

Author Daeyoung Lim [aut, cre], Ming-Hui Chen [ctb], Sungduk Kim [ctb], Joseph Ibrahim [ctb], Arvind Shah [ctb], Jianxin Lin [ctb]

Maintainer Daeyoung Lim <`Daeyoung.Lim@fda.hhs.gov`>

Repository CRAN

Date/Publication 2024-01-24 17:10:02 UTC

Contents

bayes_nmr . . . 2
bayes_parobs . . . 6
bmeta_analyze . . . 10
cholesterol . . . 14
coef.bsynthesis . . . 16
fitted.bayesnmr . . . 16
fitted.bayesparobs . . . 17
hpd . . . 17
hpd.bayesnmr . . . 18
hpd.bayesparobs . . . 19
metapack . . . 19
model_comp . . . 20
model_comp.bayesnmr . . . 20
model_comp.bayesparobs . . . 21
ns . . . 21
plot.bayesnmr . . . 22
plot.bayesparobs . . . 22
plot.sucra . . . 23
print.bayesnmr . . . 23
print.bayesparobs . . . 24
sucra . . . 24
sucra.bayesnmr . . . 25
summary.bayesnmr . . . 25
summary.bayesparobs . . . 26
TNM . . . 26

Index **28**

`bayes_nmr` *Fit Bayesian Network Meta-Regression Models*

Description

This is a function the fits the model introduced in *Bayesian Network Meta-Regression Models Using Heavy-Tailed Multivariate Random Effects with Covariate-Dependent Variances*. The first seven arguments are required except `ZCovariate`. If not provided, `ZCovariate` will be assigned a vector of ones, `rep(1, length(Outcome))`. `ZCovariate` is the centerpiece of the modeling of variances and the heavy-tailed random effects distribution.

Usage

```
bayes_nmr(
  Outcome,
  SD,
  XCovariate,
  ZCovariate,
  Treat,
  Trial,
  Npt,
  prior = list(),
  mcmc = list(),
  control = list(),
  init = list(),
  Treat_order = NULL,
  Trial_order = NULL,
  scale_x = FALSE,
  verbose = FALSE
)
```

Arguments

`Outcome`	the aggregate mean of the responses for each arm of every study.
`SD`	the standard deviation of the responses for each arm of every study.
`XCovariate`	the aggregate covariates for the fixed effects.
`ZCovariate`	the aggregate covariates associated with the variance of the random effects.
`Treat`	the treatment identifiers for trial arm. This is equivalent to the arm labels in each study. The elements within will be coerced to consecutive integers
`Trial`	the study/trial identifiers. The elements within will be coerced to consecutive integers.
`Npt`	the number of observations/participants for a unique `(k,t)`, or each arm of every trial.
`prior`	(Optional) a list of hyperparameters. The hyperparameters include `df`, `c01`, `c02`, `a4`, `b4`, `a5`, and `b5`. `df` indicates the degrees of freedom whose value is 20. The hyperparameters `a*` and `b*` will take effect only if `sample_df=TRUE`. See `control`.
`mcmc`	(Optional) a list of MCMC specification. `ndiscard` is the number of burn-in iterations. `nskip` configures the thinning of the MCMC. For instance, if `nskip=5`, `bayes_nmr` will save the posterior sample every 5 iterations. `nkeep` is the size of the posterior sample. The total number of iterations will be `ndiscard` + `nskip` * `nkeep`.
`control`	(Optional) a list of parameters for the Metropolis-Hastings algorithm. `lambda`, `phi`, and `Rho` are sampled through the localized Metropolis algorithm. `*_stepsize` with the asterisk replaced with one of the names above specifies the stepsize for determining the sample evaluation points in the localized Metropolis algorithm. `sample_Rho` can be set to `FALSE` to suppress the sampling of `Rho`. When

	sample_Rho is FALSE, Rho will be fixed using the value given by the init argument, which defaults to an equicorrelation matrix of $0.5\boldsymbol{I} + 0.5\mathbf{1}\mathbf{1}'$ where $\mathbf{1}$ is the vector of ones. When sample_df is TRUE, df will be sampled.
init	(Optional) a list of initial values for the parameters to be sampled: theta, phi, sig2, and Rho.
Treat_order	(Optional) a vector of unique treatments to be used for renumbering the Treat vector. The first element will be assigned treatment zero, potentially indicating placebo. If not provided, the numbering will default to an alphabetical/numerical order.
Trial_order	(Optional) a vector unique trials. The first element will be assigned trial zero. If not provided, the numbering will default to an alphabetical/numerical order.
scale_x	(Optional) a logical variable indicating whether XCovariate should be scaled/standardized. The effect of setting this to TRUE is not limited to merely standardizing XCovariate. The following generic functions will scale the posterior sample of theta back to its original unit: plot, fitted, summary, and print. That is theta[j] <- theta[j] / sd(XCovariate[,j]).
verbose	(Optional) a logical value indicating whether to print the progress bar during the MCMC sampling.

Value

bayes_nmr returns an object of class "bayesnmr". The functions summary or print are used to obtain and print a summary of the results. The generic accessor function fitted extracts the posterior mean, posterior standard deviation, and the interval estimates of the value returned by bayes_nmr.

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

- Outcome - the aggregate response used in the function call.
- SD - the standard deviation used in the function call.
- Npt - the number of participants for (k,t) used in the function call.
- XCovariate - the aggregate design matrix for fixed effects used in the function call. Depending on scale_x, this may differ from the matrix provided at function call.
- ZCovariate - the aggregate design matrix for random effects. bayes_nmr will assign rep(1, length(Outcome)) if it was not provided at function call.
- Trial - the *renumbered* trial indicators. Depending on Trial_order, it may differ from the vector provided at function call.
- Treat - the *renumbered* treatment indicators. Depending on Treat_order, it may differ from the vector provided at function call.
- TrtLabels - the vector of treatment labels corresponding to the renumbered Treat. This is equivalent to Treat_order if it was given at function call.
- TrialLabels - the vector of trial labels corresponding to the renumbered Trial. This is equivalent to Trial_order if it was given at function call.
- K - the total number of trials.
- nT - the total number of treatments.
- scale_x - a Boolean indicating whether XCovariate has been scaled/standardized.

- prior - the list of hyperparameters used in the function call.
- control - the list of tuning parameters used for MCMC in the function call.
- mcmctime - the elapsed time for the MCMC algorithm in the function call. This does not include all the other preprocessing and post-processing outside of MCMC.
- mcmc - the list of MCMC specification used in the function call.
- mcmc.draws - the list containing the MCMC draws. The posterior sample will be accessible here.

Author(s)

Daeyoung Lim, <daeyoung.lim@uconn.edu>

References

Li, H., Chen, M. H., Ibrahim, J. G., Kim, S., Shah, A. K., Lin, J., & Tershakovec, A. M. (2019). Bayesian inference for network meta-regression using multivariate random effects with applications to cholesterol lowering drugs. *Biostatistics*, **20(3)**, 499-516.

Li, H., Lim, D., Chen, M. H., Ibrahim, J. G., Kim, S., Shah, A. K., & Lin, J. (2021). Bayesian network meta-regression hierarchical models using heavy-tailed multivariate random effects with covariate-dependent variances. *Statistics in Medicine*.

See Also

bmeta_analyze for using the Formula interface

Examples

```
library(metapack)
data(TNM)
groupInfo <- list(c("PBO"), c("R"))
nz <- length(groupInfo)
ns <- nrow(TNM)
XCovariate <- model.matrix(~ 0 + bldlc + bhdlc + btg + age +
 white + male + bmi + potencymed + potencyhigh + durat, data = TNM)
XCovariate <- scale(XCovariate, center = TRUE, scale = FALSE)
ZCovariate <- matrix(0, ns, nz)
for (j in 1:length(groupInfo)) {
    for (i in 1:ns) {
        if (TNM$treat[i] %in% groupInfo[[j]]) {
            ZCovariate[i, j] <- 1
        }
    }
}
addz <- scale(cbind(TNM$bldlc, TNM$btg), center=TRUE, scale=TRUE)
ZCovariate <- cbind(1, ZCovariate, addz)
theta_init <- c(0.05113, -1.38866, 1.09817, -0.85855, -1.12056, -1.14133,
             -0.22435, 3.63453, -2.09322, 1.07858, 0.80566, -40.76753,
             -45.07127, -28.27232, -44.14054, -28.13203, -19.19989,
             -47.21824, -51.31234, -48.46266, -47.71443)
set.seed(2797542)
```

```
fit <- bayes_nmr(TNM$ptg, TNM$sdtg, XCovariate, ZCovariate, TNM$treat,
    TNM$trial, TNM$n, prior = list(c01 = 1.0e05, c02 = 4, df = 3),
    mcmc = list(ndiscard = 1, nskip = 1, nkeep = 1),
    init = list(theta = theta_init),
    Treat_order = c("PBO", "S", "A", "L", "R", "P", "E", "SE",
         "AE", "LE", "PE"),
    scale_x = TRUE, verbose = FALSE)
```

`bayes_parobs` *Fit Bayesian Inference for Meta-Regression*

Description

This is a function for running the Markov chain Monte Carlo algorithm for the *Bayesian inference for multivariate meta-regression with a partially observed within-study sample covariance matrix* model. The first six arguments are required. fmodel can be one of 5 numbers: 1, 2, 3, 4, and 5. The first model, fmodel = 1 denoted by M1, indicates that the Σ_{kt} are diagonal matrices with zero covariances. M2 indicates that Σ_{kt} are all equivalent but allowed to be full symmetric positive definite. M3 is where Σ_{kt} are allowed to differ across treatments, i.e., $\Sigma_{kt} = \Sigma_t$. M4 assumes thata the correlation matrix, ρ, is identical for all trials/treatments, but the variances are allowed to vary. Finally, M5 assumes a hierarchical model where $(\Sigma_{kt}|\Sigma)$ follows an inverse-Wishart distribution with fixed degrees of freedom and scale matrix Σ. Σ then follows another inverse-Wishart distribution with fixed parameters.

Usage

```
bayes_parobs(
  Outcome,
  SD,
  XCovariate,
  WCovariate,
  Treat,
  Trial,
  Npt,
  fmodel = 1,
  prior = list(),
  mcmc = list(),
  control = list(),
  init = list(),
  Treat_order = NULL,
  Trial_order = NULL,
  group = NULL,
  group_order = NULL,
  scale_x = FALSE,
  verbose = FALSE
)
```

Arguments

`Outcome`	the aggregate mean of the responses for each arm of every study.
`SD`	the standard deviation of the responses for each arm of every study.
`XCovariate`	the aggregate covariates for the fixed effects.
`WCovariate`	the aggregate covariates for the random effects.
`Treat`	the treatment identifiers. This is equivalent to the arm number of each study. The number of unique treatments must be equal across trials. The elements within will be coerced to consecutive integers.
`Trial`	the trial identifiers. This is equivalent to the arm labels in each study. The elements within will be coerced to consecutive integers
`Npt`	the number of observations/participants for a unique `(k,t)`, or each arm of every trial.
`fmodel`	the model number. The possible values for `fmodel` are 1 to 5, each indicating a different prior specification for Σ_{kt}. It will default to M1, `fmodel=1` if not specified at function call. See the following model descriptions. The objects enclosed in parentheses at the end of every bullet point are the hyperparameters associated with each model. • `fmodel=1` - $\Sigma_{kt} = diag(\sigma^2_{kt,11}, \ldots, \sigma^2_{kt,JJ})$ where $\sigma^2_{kt,jj} \sim IG(a_0, b_0)$ and $IG(a,b)$ is the inverse-gamma distribution. This specification is useful if the user does not care about the correlation recovery. (`c0`, `dj0`, `a0`, `b0`, `Omega0`) • `fmodel=2` - $\Sigma_{kt} = \Sigma$ for every combination of (k,t) and $\Sigma^{-1} \sim Wish_{s_0}(\Sigma_0)$. This specification assumes that the user has prior knowledge that the correlation structure does not change across the arms included. (`c0`, `dj0`, `s0`, `Omega0`, `Sigma0`) • `fmodel=3` - $\Sigma_{kt} = \Sigma_t$ and $\Sigma_t^{-1} \sim Wish_{s_0}(\Sigma_0)$. This is a relaxed version of `fmodel=2`, allowing the correlation structure to differ across trials but forcing it to stay identical within a trial. (`c0`, `dj0`, `s0`, `Omega0`, `Sigma0`) • `fmodel=4` - $\Sigma_{kt} = \delta_{kt}\rho\delta_{kt}$ where $\delta_{kt} = diag(\Sigma^{1/2}_{kt,11}, \ldots, \Sigma^{1/2}_{kt,JJ})$, and ρ is the correlation matrix. This specification allows the variances to vary across arms but requires that the correlations be the same. This is due to the lack of correlation information in the data, which would in turn lead to the nonidentifiability of the correlations if they were allowed to vary. However, this still is an ambitious model which permits maximal degrees of freedom in terms of variance and correlation estimation. (`c0`, `dj0`, `a0`, `b0`, `Omega0`) • `fmodel=5` - The fifth model is hierarchical and thus may require more data than the others: $(\Sigma_{kt}^{-1} \mid \Sigma) \sim Wish_{\nu_0}((\nu_0 - J - 1)^{-1}\Sigma^{-1})$ and $\Sigma \sim Wish_{d_0}(\Sigma_0)$. Σ_{kt} encodes the within-treatment-arm variation while Σ captures the between-treatment-arm variation. The hierarchical structure allows the "borrowing of strength" across treatment arms. (`c0`, `dj0`, `d0`, `nu0`, `Sigma0`, `Omega0`)
`prior`	(Optional) a list of hyperparameters. Despite `theta` in every model, each `fmodel`, along with the `group` argument, requires a different set of hyperparameters. See `fmodel` for the model specifications.

`mcmc`	(Optional) a list for MCMC specification. `ndiscard` is the number of burn-in iterations. `nskip` configures the thinning of the MCMC. For instance, if `nskip=5`, `bayes_parobs` will save the posterior sample every 5 iterations. `nkeep` is the size of the posterior sample. The total number of iterations will be `ndiscard + nskip * nkeep`.
`control`	(Optional) a list of tuning parameters for the Metropolis-Hastings algorithm. `Rho`, `R`, and `delta` are sampled through either localized Metropolis algorithm or delayed rejection robust adaptive Metropolis algorithm. `*_stepsize` with the asterisk replaced with one of the names above specifies the stepsize for determining the sample evaluation points in the localized Metropolis algorithm. `sample_Rho` can be set to `FALSE` to suppress the sampling of `Rho` for `fmodel=4`. When `sample_Rho` is `FALSE`, ρ will be fixed using the value given by the `init` argument, which defaults to $0.5I + 0.511'$ where 1 is the vector of ones.
`init`	(Optional) a list of initial values for the parameters to be sampled: `theta`, `gamR`, `Omega`, and `Rho`. The initial value for `Rho` will be effective only if `fmodel=4`.
`Treat_order`	(Optional) a vector of unique treatments to be used for renumbering the `Treat` vector. The first element will be assigned treatment zero, potentially indicating placebo. If not provided, the numbering will default to an alphabetical/numerical order.
`Trial_order`	(Optional) a vector of unique trials. The first element will be assigned zero. If not provided, the numbering will default to an alphabetical/numerical order.
`group`	(Optional) a vector containing binary variables for u_{kt}. If not provided, `bayes_parobs` will assume that there is no grouping and set $u_{kt} = 0$ for all `(k,t)`.
`group_order`	(Optional) a vector of unique group labels. The first element will be assigned zero. If not provided, the numbering will default to an alphabetical/numerical order. `group_order` will take effect only if `group` is provided by the user.
`scale_x`	(Optional) a logical variable indicating whether `XCovariate` should be scaled/standardized. The effect of setting this to `TRUE` is not limited to merely standardizing `XCovariate`. The following generic functions will scale the posterior sample of `theta` back to its original unit: `plot`, `fitted`, `summary`, and `print`.
`verbose`	(Optional) a logical variable indicating whether to print the progress bar during the MCMC sampling.

Value

`bayes_parobs` returns an object of class `"bayesparobs"`. The functions `summary` or `print` are used to obtain and print a summary of the results. The generic accessor function `fitted` extracts the posterior mean, posterior standard deviation, and the interval estimates of the value returned by `bayes_parobs`.

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

- `Outcome` - the aggregate response used in the function call.
- `SD` - the standard deviation used in the function call.
- `Npt` - the number of participants for `(k,t)` used in the function call.
- `XCovariate` - the aggregate design matrix for fixed effects used in the function call. Depending on `scale_x`, this may differ from the matrix provided at function call.

- `WCovariate` - the aggregate design matrix for random effects.
- `Treat` - the *renumbered* treatment indicators. Depending on `Treat_order`, it may differ from the vector provided at function call.
- `Trial` - the *renumbered* trial indicators. Depending on `Trial_order`, it may differ from the vector provided at function call.
- `group` - the *renumbered* grouping indicators in the function call. Depending on `group_order`, it may differ from the vector provided at function call. If `group` was missing at function call, `bayes_parobs` will assign `NULL` for `group`.
- `TrtLabels` - the vector of treatment labels corresponding to the renumbered `Treat`. This is equivalent to `Treat_order` if it was given at function call.
- `TrialLabels` - the vector of trial labels corresponding to the renumbered `Trial`. This is equivalent to `Trial_order` if it was given at function call.
- `GroupLabels` - the vector of group labels corresponding to the renumbered `group`. This is equivalent to `group_order` if it was given at function call. If `group` was missing at function call, `bayes_parobs` will assign `NULL` for `GroupLabels`.
- `K` - the total number of trials.
- `T` - the total number of treatments.
- `fmodel` - the model number as described here.
- `scale_x` - a Boolean indicating whether `XCovariate` has been scaled/standardized.
- `prior` - the list of hyperparameters used in the function call.
- `control` - the list of tuning parameters used for MCMC in the function call.
- `mcmctime` - the elapsed time for the MCMC algorithm in the function call. This does not include all the other preprocessing and post-processing outside of MCMC.
- `mcmc` - the list of MCMC specification used in the function call.
- `mcmc.draws` - the list containing the MCMC draws. The posterior sample will be accessible here.

Author(s)

Daeyoung Lim, <`daeyoung.lim@uconn.edu`>

References

Yao, H., Kim, S., Chen, M. H., Ibrahim, J. G., Shah, A. K., & Lin, J. (2015). Bayesian inference for multivariate meta-regression with a partially observed within-study sample covariance matrix. *Journal of the American Statistical Association*, **110(510)**, 528-544.

See Also

`bmeta_analyze` for using the `Formula` interface

Examples

```
library(metapack)
data("cholesterol")
Outcome <- model.matrix(~ 0 + pldlc + phdlc + ptg, data = cholesterol)
SD <- model.matrix(~ 0 + sdldl + sdhdl + sdtg, data = cholesterol)
Trial <- cholesterol$trial
Treat <- cholesterol$treat
Npt <- cholesterol$n
XCovariate <- model.matrix(~ 0 + bldlc + bhdlc + btg + age + durat +
 white + male + dm, data = cholesterol)
WCovariate <- model.matrix(~ treat, data = cholesterol)

fmodel <- 1
set.seed(2797542)
fit <- bayes_parobs(Outcome, SD, XCovariate, WCovariate, Treat, Trial,
   Npt, fmodel, mcmc = list(ndiscard = 1, nskip = 1, nkeep = 1),
   scale_x = TRUE, group = cholesterol$onstat, verbose = FALSE)
```

`bmeta_analyze` *bmeta_analyze supersedes the previous two functions: bayes_parobs, bayes_nmr*

Description

All other worker functions are superseded by this function, so that users can forget about the implementation details and focus on modeling. Meta-analytic data can be either aggregate or individual participant data (IPD). Aggregate data implies that the response consists of estimated effect sizes and their corresponding standard errors, whereas IPD is raw data. Data sets to be used for metapack should be formatted as follows:

Outcome	SD	DesignM1	DesignM2	Trial indicator (k)	Treatment indicator (t)	n
y_{13}	S_{13}	x_{13}	w_{13}	1	3	1000
y_{10}	S_{10}	x_{10}	w_{10}	1	0	545
y_{20}	S_{20}	x_{20}	w_{20}	2	0	1200

The first treatment indicator is intentionally selected to be 3, a number greater than 1, to indicate that this data format works for both meta-regression and network meta-regression. Meta-regression refers to when trials included have 2 treatments (i.e., $t = 0, 1$ for all k), and the treatments are compared head to head. On the other hand, network meta-regression includes more than two treatments, where each trial can have a different set of treatments, allowing indirect comparison between treatments that are not compared head to head as long as *consistency* holds (see Higgins et al. (2012) for consistency).

`bmeta_analyze()` and `bmeta_analyse()` are synonyms.

Usage

```
bmeta_analyze(
  formula,
  data,
  prior = list(),
  mcmc = list(),
  control = list(),
  init = list()
)

bmeta_analyse(
  formula,
  data,
  prior = list(),
  mcmc = list(),
  control = list(),
  init = list()
)
```

Arguments

`formula` an object of class Formula: a symbolic description of the meta-analytic model to fit. For aggregate models, the vector of arm sample sizes must be provided using the function `ns()`. For example, `y1 + y2 | sd1 + sd2 ~ x1 + x2 + ns(n)`—an incomplete formula only for illustration purposes. If no `ns()` is found, individual participant data (IPD) model is assumed.

`data` a data frame, list, or environment (or an object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `bmeta_analyze` is called.

`prior` an optional object that contains the hyperparameter values for the model. To see the complete list of hyperparameters for a specific model, refer to the corresponding worker function's help page, e.g., `help(bayes_parobs)` or `help(bayes_nmr)`. For meta-analysis, `model` is required in the `prior` argument, which is passed to `fmodel` as an integer. If the response is univariate, `NoRecovery` is the only valid option.

- `model="NoRecovery"` - $\Sigma_{tk} = diag(\sigma^2_{tk,11}, \ldots, \sigma^2_{tk,JJ})$ where $\sigma^2_{tk,jj} \sim IG(a_0, b_0)$ and $IG(a,b)$ is the inverse-gamma distribution. This specification is useful if the user does not care about the correlation recovery. (`c0`, `dj0`, `a0`, `b0`, `Omega0`)
- `model="EquiCovariance"` - $\Sigma_{tk} = \Sigma$ for every combination of (t,k) and $\Sigma^{-1} \sim Wish_{s_0}(\Sigma_0)$. This specification assumes that the user has prior knowledge that the correlation structure does not change across the arms included. (`c0`, `dj0`, `s0`, `Omega0`, `Sigma0`)
- `model="EquiWithinTreat"` - $\Sigma_{tk} = \Sigma_t$ and $\Sigma_t^{-1} \sim Wish_{s_0}(\Sigma_0)$. This is a relaxed version of `model=2`, allowing the correlation structure to differ

across trials but forcing it to stay identical within a trial. (c0, dj0, s0, Omega0, Sigma0)

- model="EquiCorrelation" - $\Sigma_{tk} = \delta_{tk}\rho\delta_{tk}$ where $\delta_{tk} = diag(\Sigma_{tk,11}^{1/2}, \ldots, \Sigma_{tk,JJ}^{1/2})$, and ρ is the correlation matrix. This specification allows the variances to vary across arms but requires that the correlations be the same. This is due to the lack of correlation information in the data, which would in turn lead to the nonidentifiability of the correlations if they were allowed to vary. However, this still is an ambitious model which permits maximal degrees of freedom in terms of variance and correlation estimation. (c0, dj0, a0, b0, Omega0)
- model="Hierarchical" - The fifth model is hierarchical and thus may require more data than the others: $(\Sigma_{tk}^{-1} \mid \Sigma) \sim Wish_{\nu_0}((\nu_0 - J - 1)^{-1}\Sigma^{-1})$ and $\Sigma \sim Wish_{d_0}(\Sigma_0)$. Σ_{tk} encodes the within-treatment-arm variation while Σ captures the between-treatment-arm variation. The hierarchical structure allows the "borrowing of strength" across treatment arms. (c0, dj0, d0, nu0, Sigma0, Omega0)

For network meta-analysis,

- df - the degrees of freedom of the multivariate t-distribution for the random effects. Any positive value can be assigned; if df=Inf, multivariate normal random effects will be assumed.
- c01 - the variance of the fixed-effect coefficients' prior distribuiton, a multivariate normal distribution, i.e., $\theta \sim N(0, c_1 I)$.
- c02 - the variance of the random-effects' variance-related coefficients' prior distribution, a multivariate normal distribution, i.e., $\phi \sim N(0, c_2 I)$.
- a4, b4, a5, b5 - the hyperparameters related to when the degrees of freedom for the random effects are treated as unknown/random. df is then considered to follow $Ga(\nu_a, \nu_a/\nu_b)$, $\nu_a \sim Ga(a_4, b_4)$, and $\nu_b \sim IG(a_5, b_5)$. All gamma and inverse-gamma distributions are rate-parameterized.

mcmc an optional object containing MCMC specification. ndiscard is the number of burn-in iterations. nskip configures the thinning of the MCMC. For instance, if nskip=5, parameters will be saved every 5 iterations. nkeep is the size of the posterior sample. The total number of iterations will be ndiscard + nskip * nkeep.

control an optional object that contains the control tuning parameters for the Metropolis-Hastings algorithm. Similar to prior, the complete list of control parameters for a specific model is given in the corresponding worker function's help page (see bayes_parobs or bayes_nmr). These are the lists of available tuning parameters in control for meta-analysis and network meta-analysis. Keep in mind that model will render some irrelevant tuning parameters ineffective.

- Meta-analysis - model (string), sample_Rho (logical), Rho_stepsize (double), R_stepsize (double), delta_stepsize (double), sample_Rho (logical)
- Network meta-analysis - sample_df (logical), sample_Rho (logical), lambda_stepsize (double), phi_stepsize (double), Rho_stepsize (double)

init (Optional) a list of initial values for the parameters to be sampled. The following is the list of available parameters for meta-analysis and network meta-analysis.

- Meta-analysis - `theta` (vector), `gamR` (matrix), `Omega` (matrix), `Rho` (matrix)
- Network meta-analysis - `theta` (vector), `phi` (vector), `sig2` (vector), `Rho` (matrix)

The dimensions of the initial values must be conformable for matrix operations. If dimensions don't agree, `bmeta_analyze` will tell you the correct dimension.

Details

`bmeta_analyze` currently subsumes two worker functions: `bayes_parobs` and `bayes_nmr`. `bmeta_analyze` offers a formula interface. All formulas are parsed using Formula. Formulas for `bmeta_analyze` are constrained to have a strict structure: one or two LHS, and two or three RHS. That is, `lhs_1 ~ rhs_1 | rhs2 | rhs3` or `lhs_1 | lhs_2 ~ rhs_1 | rhs2 | rhs3` (see Examples for more). The tilde (~) separates the LHS's and RHS's, each side further separated into parts by vertical bars (`|`). The meaning of each part is syntactically determined by its location inside the formula, like an English sentence. Therefore, all parts **must** come in the exact order as prescribed for `bmeta_analyze` to correctly configure your model.

- The first LHS, the responses, is required for all models.
- The second LHS is only required for aggregate models, corresponding to the standard deviations of the responses.
- The first RHS corresponds to fixed-effects covariates.
- The second RHS corresponds to the variables in either the random-effects matrix ($w'_{tk} * \gamma_k$‘) for multivariate meta-analysis or modeling the variances ($\log \tau_{tk} = z'_{tk} * \phi$) for univariate network meta-analysis.
- The third RHS corresponds to the treatment and trial indicators, and optionally the grouping variable if it exists. The order must be `treat + trial + group`, or `treat + trial` if no grouping exists. Variables here must be supplied in the exact order described; otherwise, model will not be correctly identified.

Internally, `bmeta_analyze` looks for three things: multivariate/univariate, meta-analyis/network meta-analysis, and aggregate/IPD.

- multivariate/univariate: the dimension of the response is explicit in the formula, and determines univariate versus multivariate.
- meta-analysis/network meta-analysis: the number of levels (`nlevels`) of treatments determines this. If `treat` is not already a factor variable, it is coerced to one.
- aggregate/IPD: `bmeta_analyze` looks for `ns()` in the first RHS. Aggregate models **must** provide the arm sample sizes using the function `ns()` (e.g., if `n` is the sample sizes, `y1 + y2 | sd1 + sd2 ~ x1 + x2 + ns(n)` If there is no `ns()`, IPD is assumed. Currently, IPD models are a work in progress and not supported yet.

Currently, only `univariate/multivariate` + `meta-analysis` and `univariate` + `network meta-analysis` are allowed. More models will be added in the future.

Value

`bmeta_analyze` returns a classed object of `bsynthesis` for *Bayesian synthesis*

Author(s)

Daeyoung Lim, <daeyoung.lim@uconn.edu>

References

Yao, H., Kim, S., Chen, M. H., Ibrahim, J. G., Shah, A. K., & Lin, J. (2015). Bayesian inference for multivariate meta-regression with a partially observed within-study sample covariance matrix. *Journal of the American Statistical Association*, **110(510)**, 528-544.

Li, H., Chen, M. H., Ibrahim, J. G., Kim, S., Shah, A. K., Lin, J., & Tershakovec, A. M. (2019). Bayesian inference for network meta-regression using multivariate random effects with applications to cholesterol lowering drugs. *Biostatistics*, **20(3)**, 499-516.

Li, H., Lim, D., Chen, M. H., Ibrahim, J. G., Kim, S., Shah, A. K., & Lin, J. (2021). Bayesian network meta-regression hierarchical models using heavy-tailed multivariate random effects with covariate-dependent variances. *Statistics in Medicine*.

See Also

bayes_parobs for multivariate meta-analysis, and bayes_nmr for univariate network meta-analysis.

Examples

```
set.seed(2797542)
data("cholesterol")
f_1 <- 'pldlc + phdlc + ptg | sdldl + sdhdl + sdtg ~ 0 + bldlc + bhdlc + btg +
  age + durat + white + male + dm + ns(n) | treat | treat + trial + onstat'
out_1 <- bmeta_analyze(as.formula(f_1), data = cholesterol,
  prior = list(model="NoRecovery"),
  mcmc = list(ndiscard = 3, nskip = 1, nkeep = 1),
  control=list(scale_x = TRUE, verbose=FALSE))

set.seed(2797542)
data("TNM")
TNM$group <- factor(match(TNM$treat, c("PBO", "R"), nomatch = 0))
f_2 <- 'ptg | sdtg ~
  0 + bldlc + bhdlc + btg + age + white + male + bmi +
  potencymed + potencyhigh + durat + ns(n) |
  scale(bldlc) + scale(btg) + group | treat  + trial'
out_2 <- bmeta_analyze(as.formula(f_2), data = TNM,
  mcmc = list(ndiscard = 1, nskip = 1, nkeep = 1),
  control=list(scale_x = TRUE, verbose=FALSE))
```

---

cholesterol — *26 double-blind, randomized, active, or placebo-controlled clinical trials on patients with primary hypercholesterolemia sponsored by Merck & Co., Inc., Kenilworth, NJ, USA.*

Description

A data set containing clinical trial on hypercholesterolemia including 26 trials and 2 treatment arms each, and other attributes of the participants

Usage

```
data(cholesterol)
```

Format

A data frame with 52 rows and 19 variables

study study identifier

trial trial identifier

treat treatment indicator for Statin or Statin+Ezetimibe

n the number of participants in the study arms corresponding to the trial and treatment

pldlc aggregate percentage change in LDL-C

phdlc aggregate percentage change from baseline in HDL-C

ptg aggregate percentage change from baseline in triglycerides (TG)

sdldl sample standard deviation of percentage change in LDL-C

sdhdl sample standard deviation of percentage change in HDL-C

sdtg sample standard deviation of percentage change in triglycerides (TG)

onstat whether the participants were on Statin prior to the trial

bldlc baseline LDL-C

bhdlc baseline HDL-C

btg baseline triglycerides (TG)

age age in years

white the proportion of white participants

male the proportion of male participants

dm the proportion of participants with diabetes mellitus

durat duration in weeks

Examples

```
data(cholesterol)
```

coef.bsynthesis *get the posterior mean of fixed-effect coefficients*

Description

get the posterior mean of fixed-effect coefficients

Usage

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

Arguments

`object`	a class of `bsynthesize`
`...`	other arguments

Value

Coefficients extracted from the model object `object`

fitted.bayesnmr *get fitted values*

Description

get fitted values

Usage

```
## S3 method for class 'bayesnmr'
fitted(object, level = 0.95, HPD = TRUE, ...)
```

Arguments

`object`	the output model from fitting a meta analysis/regression model
`level`	credible level for interval estimation; set to 0.95 by default
`HPD`	a logical argument indicating whether HPD intervals should be computed; if FALSE, equal-tail credible intervals are computed
`...`	additional arguments for fitted

Value

a list of fitted values

`fitted.bayesparobs` *get fitted values*

Description

get fitted values

Usage

```
## S3 method for class 'bayesparobs'
fitted(object, level = 0.95, HPD = TRUE, ...)
```

Arguments

`object`	the output model from fitting a meta analysis/regression model
`level`	credible level for interval estimation; set to 0.95 by default
`HPD`	a logical argument indicating whether HPD intervals should be computed; if FALSE, equal-tail credible intervals are computed
`...`	additional arguments for fitted

Value

a list of fitted values

`hpd` *get the highest posterior density (HPD) interval*

Description

get the highest posterior density (HPD) interval

Usage

```
hpd(object, parm, level = 0.95, HPD = TRUE)
```

Arguments

`object`	the output model from fitting a (network) meta analysis/regression model
`parm`	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
`level`	the probability which the HPD interval will cover
`HPD`	a logical value indicating whether HPD or equal-tailed credible interval should be computed; by default, TRUE

Details

A $100(1-\alpha)\%$ HPD interval for θ is given by

$$R(\pi_\alpha) = \theta : \pi(\theta|D) \geq \pi_\alpha,$$

where π_α is the largest constant that satisfies $P(\theta \in R(\pi_\alpha)) \geq 1-\alpha$. hpd computes the HPD interval from an MCMC sample by letting $\theta_{(j)}$ be the jth smallest of the MCMC sample, θ_i and denoting

$$R_j(n) = (\theta_{(j)}, \theta_{(j+[(1-\alpha)n])}),$$

for $j = 1, 2, \ldots, n-[(1-\alpha)n]$. Once θ_i's are sorted, the appropriate j is chosen so that

$$\theta_{(j+[(1-\alpha)n])} - \theta_{(j)} = \min_{1\leq j\leq n-[(1-\alpha)n]} (\theta_{(j+[(1-\alpha)n])} - \theta_{(j)}).$$

Value

dataframe containing HPD intervals for the parameters

References

Chen, M. H., & Shao, Q. M. (1999). Monte Carlo estimation of Bayesian credible and HPD intervals. *Journal of Computational and Graphical Statistics*, **8(1)**, 69-92.

---

hpd.bayesnmr *get the highest posterior density (HPD) interval*

Description

get the highest posterior density (HPD) interval

Usage

```
## S3 method for class 'bayesnmr'
hpd(object, parm, level = 0.95, HPD = TRUE)
```

Arguments

object	the output model from fitting a (network) meta analysis/regression model
parm	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
level	the probability which the HPD interval will cover
HPD	a logical value indicating whether HPD or equal-tailed credible interval should be computed; by default, TRUE

Value

dataframe containing HPD intervals for the parameters `hpd.bayesparobs` *get the highest posterior density (HPD) interval or equal-tailed credible interval*

Description

get the highest posterior density (HPD) interval or equal-tailed credible interval

Usage

```
## S3 method for class 'bayesparobs'
hpd(object, parm, level = 0.95, HPD = TRUE)
```

Arguments

`object`	the output model from fitting a (network) meta analysis/regression model
`parm`	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
`level`	the probability which the HPD interval will cover
`HPD`	a logical value indicating whether HPD or equal-tailed credible interval should be computed; by default, TRUE

Value

dataframe containing HPD intervals for the parameters

`metapack` *metapack: a package for Bayesian meta-analysis and network meta-analysis*

Description

The metapack package provides one category of functions: bayes.parobs and bayes.nmr

Multivariate Meta-Regression function

The bayes.parobs function fits the multivariate meta-regression model with partially observed sample covariance matrix to the given data.

Network Meta-Regression function

The bayes.nmr function fits the network meta-regression model with heavy-tailed random effects distribution to the given data.

model_comp — *compute the model comparison measures: DIC, LPML, or Pearson's residuals*

Description

`model_comp` is a generic function that computes the model comparison measures (DIC and LPML) or the Pearson's residuals. Note that the Pearson's residuals are not available for `bayes.nmr` when `df` is either random or fixed but smaller than 2 since the variance of the random effects is not finite.

Usage

```
model_comp(object, type = "lpml", verbose = FALSE, ncores = NULL)
```

Arguments

`object`	the output model from fitting a meta analysis/regression model
`type`	the type of model comparison measure to compute; DIC or LPML
`verbose`	FALSE by default; If TRUE, then progress bar will appear
`ncores`	the number of CPU cores to use for parallel processing. It must not exceed the number of existing cores. If unspecified, it will default to 2 cores or the number of existing cores, whichever is smaller.

Value

dataframe containing the compute the model comparison measures

model_comp.bayesnmr — *get compute the model comparison measures*

Description

get compute the model comparison measures

Usage

```
## S3 method for class 'bayesnmr'
model_comp(object, type = "lpml", verbose = FALSE, ncores = NULL)
```

Arguments

`object`	the output model from fitting a meta analysis/regression model
`type`	the type of model comparison measures; DIC or LPML
`verbose`	FALSE by default; If TRUE, then progress bar will appear
`ncores`	the number of CPU cores to use for parallel processing. It must not exceed the number of existing cores. If unspecified, it will default to 2 cores or the number of existing cores, whichever is smaller.

Value

dataframe containing the compute the model comparison measures

---

`model_comp.bayesparobs` *compute the model comparison measures*

Description

compute the model comparison measures

Usage

```
## S3 method for class 'bayesparobs'
model_comp(object, type = "lpml", verbose = FALSE, ncores = NULL)
```

Arguments

`object`	the output model from fitting a meta analysis/regression model
`type`	the type of model comparison measures; DIC or LPML
`verbose`	FALSE by default; If TRUE, then progress bar will appear
`ncores`	the number of CPU cores to use for parallel processing. It must not exceed the number of existing cores. If unspecified, it will default to 2 cores or the number of existing cores, whichever is smaller.

Value

dataframe containing the compute the model comparison measures

---

`ns` *helper function encoding trial sample sizes in formulas*

Description

helper function encoding trial sample sizes in formulas

Usage

```
ns(x)
```

Arguments

`x`	the name of the variable containing trial sample sizes

plot.bayesnmr *get goodness of fit*

Description

get goodness of fit

Usage

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

Arguments

x	the output model from fitting a meta analysis/regression model
...	additional parameters for plot

Value

No return value

plot.bayesparobs *get goodness of fit*

Description

get goodness of fit

Usage

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

Arguments

x	the output model from fitting a meta analysis/regression model
...	additional parameters for plot

Value

No return value

plot.sucra *plot the surface under the cumulative ranking curve (SUCRA)*

Description

plot the surface under the cumulative ranking curve (SUCRA)

Usage

```
## S3 method for class 'sucra'
plot(x, legend.position = "none", ...)
```

Arguments

`x`	the output model from fitting a network meta analysis/regression model
`legend.position`	the position of the legend that will be passed onto ggplot
`...`	additional arguments for plot

Value

No return value

print.bayesnmr *Print results*

Description

Print results

Usage

```
## S3 method for class 'bayesnmr'
print(x, level = 0.95, HPD = TRUE, ...)
```

Arguments

`x`	the output model from fitting a network meta analysis/regression model
`level`	credible level for interval estimation; set to 0.95 by default
`HPD`	a logical argument indicating whether HPD intervals should be computed; if FALSE, equal-tail credible intervals are computed
`...`	additional arguments for print

Value

No return value; print a summary of the output

`print.bayesparobs` *Print results*

Description

Print results

Usage

```
## S3 method for class 'bayesparobs'
print(x, level = 0.95, HPD = TRUE, ...)
```

Arguments

`x`	the output model from fitting a meta analysis/regression model
`level`	credible level for interval estimation; set to 0.95 by default
`HPD`	a logical argument indicating whether HPD intervals should be computed; if FALSE, equal-tail credible intervals are computed
`...`	additional arguments for print

Value

No return value; print a summary of the output

`sucra` *get surface under the cumulative ranking curve (SUCRA)*

Description

get surface under the cumulative ranking curve (SUCRA)

Usage

```
sucra(object)
```

Arguments

`object`	the output model from fitting a network meta analysis/regression model

Value

a list containing SUCRA and the discrete rank probability matrix of size T by T

`sucra.bayesnmr` *get surface under the cumulative ranking curve (SUCRA)*

Description

get surface under the cumulative ranking curve (SUCRA)

Usage

```
## S3 method for class 'bayesnmr'
sucra(object)
```

Arguments

`object`	the output model from fitting a network meta analysis/regression model

Value

a list containing SUCRA and the discrete rank probability matrix of size T by T

`summary.bayesnmr` *‘summary‘ method for class "‘bayesnmr‘"*

Description

‘summary‘ method for class "‘bayesnmr‘"

Usage

```
## S3 method for class 'bayesnmr'
summary(object, level = 0.95, HPD = TRUE, ...)
```

Arguments

`object`	the output model from fitting a network meta analysis/regression model
`level`	credible level for interval estimation; set to 0.95 by default
`HPD`	a logical argument indicating whether HPD intervals should be computed; if FALSE, equal-tail credible intervals are computed
`...`	additional arguments for print

Value

does not return anything; print a summary of the output

summary.bayesparobs — summary *method for class* "bayesparobs"

Description

summary method for class "bayesparobs"

Usage

```
## S3 method for class 'bayesparobs'
summary(object, level = 0.95, HPD = TRUE, ...)
```

Arguments

`object`	the output model from fitting a meta analysis/regression model
`level`	credible level for interval estimation; set to 0.95 by default
`HPD`	a logical argument indicating whether HPD intervals should be computed; if FALSE, equal-tail credible intervals are computed
`...`	additional arguments for summary

Value

print summary for the model fit

TNM — *Triglycerides Network Meta (TNM) data*

Description

A systemically reviewed network meta data set on tryglyceride (TG) lowering drugs

Usage

```
data(TNM)
```

Format

A data frame with 73 rows and 15 variables

trial trial identifier

treat treatment indicator for placebo (PBO), simvastatin (S), atorvastatin (A), lovastatin (L), rosuvastatin (R), pravastatin (P), ezetimibe (E), simvastatin+ezetimibe (SE), atorvastatin+ezetimibe (AE), lovastatin+ezetimibe (LE), or pravastatin+ezetimibe (PE)

n the number of participants in the study corresponding to the trial and treatment

ptg percentage change from baseline in triglycerides (TG)

sdtg sample standard deviation of percentage change in triglycerides (TG)

bldlc baseline LDL-C

bhdlc baseline HDL-C

btg baseline triglycerides (TG)

age age in years

white the proportion of white participants

male the proportion of male participants

bmi body fat index

potencymed the proportion of medium statin potency

potencyhigh the proportion of high statin potency

durat duration in weeks

Examples

```
data(TNM)
```

Index

∗ **datasets**
  cholesterol, 14
  TNM, 26

as.data.frame, *11*

bayes_nmr, 2, *12*, *14*
bayes_parobs, 6, *12*, *14*
bmeta_analyse *(*bmeta_analyze*)*, 10
bmeta_analyze, *5*, *9*, 10

cholesterol, 14
coef.bsynthesis, 16

fitted.bayesnmr, 16
fitted.bayesparobs, 17
Formula, *5*, *9*, *11*, *13*

hpd, 17
hpd.bayesnmr, 18
hpd.bayesparobs, 19

metapack, 19
model_comp, 20
model_comp.bayesnmr, 20
model_comp.bayesparobs, 21

ns, 21

plot.bayesnmr, 22
plot.bayesparobs, 22
plot.sucra, 23
print.bayesnmr, 23
print.bayesparobs, 24

sucra, 24
sucra.bayesnmr, 25
summary.bayesnmr, 25
summary.bayesparobs, 26

TNM, 26