## Package ‘bipd’

March 2, 2022

**Type** Package  
**Title** Bayesian Individual Patient Data Meta-Analysis using 'JAGS'  
**Version** 0.2  
**Date** 2022-03-02  
**Depends** R (>= 2.10)  
**Imports** rjags (>= 4-6), coda (>= 0.13), mvtnorm, dplyr  
**Suggests** dclone, R2WinBUGS, mice, micemd, miceadds, mitools, knitr, rmarkdown  
**Description**  
We use a Bayesian approach to run individual patient data meta-analysis and network meta-analysis using 'JAGS'. The methods incorporate shrinkage methods and calculate patient-specific treatment effects as described in Seo et al. (2021) [DOI:10.1002/sim.8859]. This package also includes user-friendly functions that impute missing data in an individual patient data using mice-related packages.  
**License** GPL-3  
**Encoding** UTF-8  
**RoxygenNote** 7.1.2  
**VignetteBuilder** knitr  
**NeedsCompilation** no  
**Author** Michael Seo [aut, cre]  
**Maintainer** Michael Seo <swj8874@gmail.com>  
**Repository** CRAN  
**Date/Publication** 2022-03-02 15:30:05 UTC

### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Rmarkdown</th>
</tr>
</thead>
<tbody>
<tr>
<td>bipd-package</td>
<td>2</td>
</tr>
<tr>
<td>findMissingPattern</td>
<td>3</td>
</tr>
<tr>
<td>generate_ipdma_example</td>
<td>4</td>
</tr>
<tr>
<td>generate_ipdnma_example</td>
<td>4</td>
</tr>
</tbody>
</table>
bipd-package

A package for individual patient data meta-analysis using 'JAGS'

Details

We use a Bayesian approach to run individual patient data meta-analysis and network meta-analysis using 'JAGS'. The methods incorporate shrinkage methods and calculate patient-specific treatment effects as described in Seo et al. (2021) <DOI:10.1002/sim.8859>. This package also includes user-friendly functions that impute missing data in an individual patient data using mice-related packages.

References


findMissingPattern

Find missing data pattern in a given data

Description

Find missing data pattern in a given data i.e. whether variables are systematically missing or sporadically missing. Also calculates missing count and percentage for exploratory purposes.

Usage

findMissingPattern(
  dataset = NULL,
  covariates = NULL,
  typeofvar = NULL,
  studyname = NULL,
  treatmentname = NULL,
  outcomename = NULL
)

Arguments

dataset data which contains variables of interests
covariates vector of variable names that the user is interested in finding a missing data pattern
typeofvar type of covariate variables; should be a vector of these values: "continuous", "binary", or "count". Order should follow that of covariates parameter.
studyname study name in the data specified
treatmentname treatment name in the data specified
outcomename outcome name in the data specified

Value

missingcount missing number of patients for each study and covariate
missingpercent missing percentage of patients for each study and covariate
sys_missing a vector indicating whether each covariate is systematically missing
spor_missing a vector indicating whether each covariate is sporadically missing
sys_covariates a vector of systematically missing covariates
spor_covariates a vector of sporadically missing covariates
without_sys_covariates a vector of covariates that are not systematically missing
covariates vector of variable names that the user is interested in finding a missing data pattern
studyname study name in the data specified
treatmentname treatment name in the data specified
outcomename outcome name in the data specified
Examples

```r
simulated_dataset <- generate_sysmiss_ipdma_example(Nstudies = 10, Ncov = 5, sys_missing_prob = 0.3, magnitude = 0.2, heterogeneity = 0.1)

missP <- findMissingPattern(simulated_dataset, covariates = c("x1", "x2", "x3", "x4", "x5"), typeofvar = c("continuous", "binary", "binary", "continuous", "continuous"), studynam = "study", treatmentname = "treat", outcomename = "y")

missP
```

---

**generate_ipdma_example**

*Generate a simulated IPD-MA data for demonstration*

**Description**

Generate a simulated IPD-MA data for demonstration

**Usage**

```r
generate_ipdma_example(type = "continuous")
```

**Arguments**

- `type`: "continuous" for continuous outcome and "binary" for binary outcome

**Value**

returns simulated IPD-MA data

**Examples**

```r
ds <- generate_ipdma_example(type = "continuous")
head(ds)
```

---

**generate_ipdnma_example**

*Generate a simulated IPD-NMA data for demonstration*

**Description**

Generate a simulated IPD-NMA data for demonstration

**Usage**

```r
generate_ipdnma_example(type = "continuous")
```
Arguments

- type: "continuous" for continuous outcome and "binary" for binary outcome

Value

- return simulated IPD-NMA data

Examples

```r
simulated_dataset <- generate_sysmiss_ipdma_example(Nstudies = 10, Ncv = 5, sys_missing_prob = 0.3, magnitude = 0.2, heterogeneity = 0.1)
head(simulated_dataset)
```
ipd.run

Run the model using the ipd object

Description
This is the core function that runs the model in our program. Before running this function, we need to specify data, prior, JAGS code, etc. using ipd.model type function.

Usage
ipd.run(
ipd,
pars.save = c("beta", "gamma", "delta"),
inits = NULL,
n.chains = 3,
n.adapt = 1000,
n.burnin = 1000,
n.iter = 1000)

Arguments
ipd ipd object created from ipd.model type function
pars.save parameters to save. For instance, "beta" - coefficients for main effects; "gamma" - coefficients for effect modifiers; "delta" - average treatment effect
inits initial values specified for the parameters to save
n.chains number of MCMC chains to sample
n.adapt number of iterations for adaptation (Note that the samples from adaptation phase is non-Markovian and do not constitute a Markov chain)
n.burnin number of iterations for burn-in
n.iter number of iterations to run after the adaptation

Value
MCMC samples stored using JAGS. The returned samples have the form of mcmc.list and coda functions can be directly applied.

Examples
ds <- generate_ipdma_example(type = "continuous")
ipd <- with(ds, ipdma.model.onestage(y = y, study = studyid, treat = treat, X = cbind(z1, z2), response = "normal", shrinkage = "none"))
samples <- ipd.run(ipd, pars.save = c("beta", "gamma", "delta"), n.chains = 3, n.burnin = 500, n.iter = 5000)
ipd.run.parallel  Run the model using the ipd object with parallel computation

Description

This function runs the model through parallel computation using dclone R package. Before running this function, we need to specify data, prior, JAGS code, etc. using ipd.model type function.

Usage

ipd.run.parallel(
  ipd,
  pars.save = c("beta", "gamma", "delta"),
  inits = NULL,
  n.chains = 3,
  n.adapt = 1000,
  n.burnin = 1000,
  n.iter = 10000
)

Arguments

ipd ipd object created from ipd.model type function
pars.save parameters to save. For instance, "beta" - coefficients for main effects; "gamma" - coefficients for effect modifiers; "delta" - average treatment effect
inits initial values specified for the parameters to save
n.chains number of MCMC chains to sample
n.adapt number of iterations for adaptation (Note that the samples from adaptation phase is non-Markovian and do not constitute a Markov chain)
n.burnin number of iterations for burn-in
n.iter number of iterations to run after the adaptation

Value

MCMC samples stored using JAGS. The returned samples have the form of mcmc.list and coda functions can be directly applied.

Examples

ds <- generate_ipdma_example(type = "continuous")
ipd <- with(ds, ipdma.model.onestage(y = y, study = studyid, treat = treat, X = cbind(z1, z2), response = "normal", shrinkage = "none"))
samples <- ipd.run.parallel(ipd, pars.save = c("beta", "gamma", "delta"), n.chains = 3, n.burnin = 500, n.iter = 5000)
ipdma.impute

Impute missing data in individual participant data with two treatments (i.e. placebo and a treatment).

Description

Impute missing data in individual participant data with two treatments. Data is clustered by different studies. In the presence of systematically missing variables, the function defaults to 2l.2stage.norm, 2l.2stage.bin, and 2l.2stage.pois methods in micemd package. If there are no systematically missing variables, the function defaults to use 2l.pmm in miceadds package which generalizes predictive mean matching using linear mixed model. If there is only one study available, the function defaults to use pmm in mice package.

Usage

ipdma.impute(
  dataset = NULL,
  covariates = NULL,
  typeofvar = NULL,
  sys_impute_method = "2l.2stage",
  interaction = NULL,
  meth = NULL,
  pred = NULL,
  studyname = NULL,
  treatmentname = NULL,
  outcomename = NULL,
  m = 5
)

Arguments

dataset data which contains variables of interests
covariates vector of variable names to find missing data pattern
typeofvar type of covariate variables; should be a vector of these values: "continuous", "binary", or "count". Order should follow that of covariates parameter specified. Covariates that are specified "binary" are automatically factored.
sys_impute_method method used for systematically missing studies. Options are "2l.glm", "2l.2stage", or "2l.jomo". Default is set to "2l.2stage". There is also an option to ignore all the clustering level and impute using predictive mean matching by setting this parameter to "pmm".
interaction indicator denoting whether treatment-covariate interactions should be included. Default is set to true.
meth imputation method to be used in the mice package. If left unspecified, function picks a reasonable one.
pred correct prediction matrix to be used in the mice package. If left unspecified, function picks a reasonable one.
studyname study name in the data specified.
treatmentname treatment name in the data specified.
outcome name in the data specified.
m number of imputed datasets. Default is set to 5.

Value

missingPattern missing pattern object returned by running findMissingPattern function
meth imputation method used with the mice function
pred prediction matrix used with the mice function
imp imputed datasets that is returned from the mice function
imp.list imputed datasets in a list format

Examples

# load in mice packages
# library(mice) #for datasets with only one study level
# library(miceadds) #for multilevel datasets without systematically missing predictors
# library(micemd) #for multilevel datasets with systematically missing predictors.
simulated_dataset <- generate_sysmiss_ipdma_example(Nstudies = 10, Ncov = 5, sys_missing_prob = 0.3,
magnitude = 0.2, heterogeneity = 0.1)

imputation <- ipdma.impute(simulated_dataset, covariates = c("x1", "x2", "x3", "x4", "x5"),
typeofvar = c("continuous", "binary", "binary", "continuous", "continuous"), interaction = TRUE,
studyname = "study", treatmentname = "treat", outcome name = "y", m = 5)
response = "normal",
    type = "random",
    mean.alpha = 0,
    prec.alpha = 0.001,
    mean.beta = 0,
    prec.beta = 0.001,
    mean.gamma.within = 0,
    prec.gamma.within = 0.001,
    mean.gamma.across = 0,
    prec.gamma.across = 0.001,
    mean.delta = 0,
    prec.delta = 0.001,
    hy.prior = list("dhnorm", 0, 1)
)

Arguments

y                outcome of the study. Can be continuous or binary.
study            vector indicating which study the patient belongs to. Please change the study
                 names into numbers (i.e. 1, 2, 3, etc)
treat            vector indicating which treatment the patient was assigned to (i.e. 1 for treat-
                 ment, 0 for placebo)
X                matrix of covariate values for each patient. Dimension would be number of
                 patients x number of covariates.
response         specification of the outcome type. Must specify either "normal" or "binomial".
type             assumption on the treatment effect: either "random" for random effects model
                 or "fixed" for fixed effects model. Default is "random".
mean.alpha        prior mean for the study intercept
prec.alpha        prior precision for the study intercept
mean.beta         prior mean for the regression coefficients of the main effects of the covariates;
                 main effects are assumed to have common effect.
prec.beta         prior precision for the regression coefficients of the main effects of the covariates
mean.gamma.within prior mean for effect modifiers of within study information.
prec.gamma.within prior precision for the effect modifiers of within study information.
mean.gamma.across prior mean for the effect modifiers of across study information; effect modifica-
                 tion is assumed to have common effect.
prec.gamma.across prior precision for the effect modifiers of across study information
mean.delta        prior mean for the average treatment effect
prec.delta        prior precision for the average treatment effect
ipdma.model.onestage

**Value**

- **data.JAGS**: data organized in a list so that it can be used when running code in JAGS
- **code**: JAGS code that is used to run the model. Use cat(code) to see the code in a readable format
- **model.JAGS**: JAGS code in a function. This is used when running model in parallel
- **Xbar**: study specific averages of covariates

**References**


**Examples**

```r
ds <- generate_ipdma_example(type = "continuous")
ipd <- with(ds, ipdma.model.deft.onestage(y = y, study = studyid, treat = treat, X = cbind(z1, z2), response = "normal"))
cat(ipd$code)
samples <- ipd.run(ipd, pars.save = c("beta", "gamma.within", "gamma.across", "delta"))
treatment.effect(ipd, samples, newpatient= c(1,0.5), reference = c(0, 0))
```

---

**ipdma.model.onestage**  
*Make an one-stage individual patient data meta-analysis object containing data, priors, and a JAGS model code*

**Description**

This function sets up data and JAGS code that is needed to run one-stage IPD-MA models in JAGS.

**Usage**

```r
ipdma.model.onestage(
  y = NULL,
  study = NULL,
  treat = NULL,
  X = NULL,
  response = "normal",
)```
type = "random",
shrinkage = "none",
scale = TRUE,
mean.alpha = 0,
prec.alpha = 0.001,
mean.beta = 0,
prec.beta = 0.001,
mean.gamma = 0,
prec.gamma = 0.001,
mean.delta = 0,
prec.delta = 0.001,
hy.prior = list("dnorm", 0, 1),
lambda.prior = NULL,
p.ind = NULL,
g = NULL,
hy.prior.eta = NULL
)

Arguments

y
outcome of the study. Can be continuous or binary.

study
vector indicating which study the patient belongs to. Please change the study names into numbers (i.e. 1, 2, 3, etc)

treat
vector indicating which treatment the patient was assigned to (i.e. 1 for treatment, 0 for placebo)

X
matrix of covariate values for each patient. Dimension would be number of patients x number of covariates.

response
specification of the outcome type. Must specify either "normal" or "binomial".

type
assumption on the treatment effect: either "random" for random effects model or "fixed" for fixed effects model. Default is "random".

shrinkage
shrinkage method applied to the effect modifiers. "none" correspond to no shrinkage. "laplace" corresponds to a adaptive shrinkage with a Laplacian prior (ie often known as Bayesian LASSO). "SSVS" corresponds to the Stochastic Search Variable Selection method. SSVS is not strictly a shrinkage method, but pulls the estimated coefficient toward zero through variable selection in each iteration of the MCMC. See O’hara et al (2009) for more details.

scale
indicator for scaling the covariates by the overall average; default is TRUE.

mean.alpha
prior mean for the study intercept

prec.alpha
prior precision for the study intercept

mean.beta
prior mean for the regression coefficients of the main effects of the covariates; main effects are assumed to have common effect.

prec.beta
prior precision for the regression coefficients of the main effects of the covariates

mean.gamma
prior mean for the effect modifiers. This parameter is not used if penalization is placed on effect modifiers.
prec.gamma  prior precision for the effect modifiers. This parameter is not used if penalization is placed on effect modifiers.

mean.delta  prior mean for the average treatment effect

prec.delta  prior precision for the average treatment effect

hy.prior  prior for the heterogeneity parameter. Supports uniform, gamma, and half normal for normal and binomial response. It should be a list of length 3, where first element should be the distribution (one of dunif, dgamma, dhnorm) and the next two are the parameters associated with the distribution. For example, list("dunif", 0, 5) gives uniform prior with lower bound 0 and upper bound 5 for the heterogeneity parameter.

lambda.prior  (only for shrinkage = "laplace") two options for laplace shrinkage. We can put a gamma prior on the lambda (i.e. list("dgamma",2,0.1)) or put a uniform prior on the inverse of lambda (i.e. list("dunif",0,5))

p.ind  (only for shrinkage = "SSVS") prior probability of including each of the effect modifiers. Length should be same as the total length of the covariates.

g  (only for shrinkage = "SSVS") multiplier for the precision of spike. Default is g = 1000.

hy.prior.eta  (only for shrinkage = "SSVS") standard deviation of the slab prior. Currently only support uniform distribution. Default is list("dunif", 0, 5)

Value

data.JAGS  data organized in a list so that it can be used when running code in JAGS

code  JAGS code that is used to run the model. Use cat(code) to see the code in a readable format

model.JAGS  JAGS code in a function. This is used when running model in parallel

scale.mean  mean used in scaling covariates

scale.sd  standard deviation used in scaling covariates

References


Examples

ds <- generate_ipdma_example(type = "continuous")
ipd <- with(ds, ipdma.model.onestage(y = y, study = studyid, treat = treat, X = cbind(z1, z2), response = "normal", shrinkage = "none"))
ipdnma.model.onestage Make an one-stage individual patient data network meta-analysis object containing data, priors, and a JAGS model code

Description

This function sets up data and JAGS code that is needed to run one-stage IPD-NMA models in JAGS.

Usage

```r
ipdnma.model.onestage(
  y = NULL,
  study = NULL,
  treat = NULL,
  X = NULL,
  response = "normal",
  type = "random",
  shrinkage = "none",
  scale = TRUE,
  mean.alpha = 0,
  prec.alpha = 0.001,
  mean.beta = 0,
  prec.beta = 0.001,
  mean.gamma = 0,
  prec.gamma = 0.001,
  mean.delta = 0,
  prec.delta = 0.001,
  hy.prior = list("dhnorm", 0, 1),
  lambda.prior = NULL,
  p.ind = NULL,
  g = NULL,
  hy.prior.eta = NULL
)
```

Arguments

- `y`: outcome of the study. Can be continuous or binary.
- `study`: vector indicating which study the patient belongs to. Please change the study names into numbers (i.e. 1, 2, 3, etc).
- `treat`: vector indicating which treatment the patient was assigned to. Since this is a network meta-analysis and there would be more than 2 treatments, careful naming of treatment is needed. This vector needs to be a sequence from 1:NT where NT is the total number of treatments. Treatment that is assigned 1 would be the baseline treatment.
matrix of covariate values for each patient. Dimension would be number of patients x number of covariates.

response specification of the outcome type. Must specify either "normal" or "binomial".

type assumption on the treatment effect: either "random" for random effects model or "fixed" for fixed effects model. Default is "random".

shrinkage shrinkage method applied to the effect modifiers. "none" correspond to no shrinkage. "laplace" corresponds to a adaptive shrinkage with a Laplacian prior (ie often known as Bayesian LASSO). "SSVS" corresponds to the Stochastic Search Variable Selection method. SSVS is not strictly a shrinkage method, but pulls the estimated coefficient toward zero through variable selection in each iteration of the MCMC. See O’hara et al (2009) for more details.

scale indicator for scaling the covariates by the overall average; default is TRUE.

mean.alpha prior mean for the study intercept

prec.alpha prior precision for the study intercept

mean.beta prior mean for the regression coefficients of the main effects of the covariates; main effects are assumed to have common effect.

prec.beta prior precision for the regression coefficients of the main effects of the covariates

mean.gamma prior mean for the effect modifiers. This parameter is not used if penalization is placed on effect modifiers.

prec.gamma prior precision for the effect modifiers. This parameter is not used if penalization is placed on effect modifiers.

mean.delta prior mean for the average treatment effect

prec.delta prior precision for the average treatment effect

hy.prior prior for the heterogeneity parameter. Supports uniform, gamma, and half normal for normal and binomial response It should be a list of length 3, where first element should be the distribution (one of dunif, dgamma, dhnorm) and the next two are the parameters associated with the distribution. For example, list("dunif", 0, 5) gives uniform prior with lower bound 0 and upper bound 5 for the heterogeneity parameter.

lambda.prior (only for shrinkage = "laplace") two options for laplace shrinkage. We can put a gamma prior on the lambda (i.e. list("dgamma",2,0.1)) or put a uniform prior on the inverse of lambda (i.e. list("dunif",0,5))

p.ind (only for shrinkage = "SSVS") prior probability of including each of the effect modifiers. Length should be same as the total length of the covariates.

g (only for shrinkage = "SSVS") multiplier for the precision of spike. Default is g = 1000.

hy.prior.eta (only for shrinkage = "SSVS") standard deviation of the slab prior. Currently only support uniform distribution. Default is list("dunif", 0, 5)

Value

data.JAGS data organized in a list so that it can be used when running code in JAGS

code JAGS code that is used to run the model. Use cat(code) to see the code in a readable format
model.JAGS JAGS code in a function. This is used when running model in parallel
scale.mean mean used in scaling covariates
scale.sd standard deviation used in scaling covariates

References


Examples
ds <- generate_ipdnma_example(type = "continuous")
ipd <- with(ds, ipdnma.model.onestage(y = y, study = studyid, treat = treat, X = cbind(z1, z2), response = "normal", shrinkage = "none"))

treatment.effect Calculate patient-specific treatment effect

Description

Function for calculating the patient-specific treatment effect. Patient-specific treatment effect includes the main effect of treatment and treatment-covariate interaction effect (i.e. effect modification). Reports odds ratio for the binary outcome.

Usage
treatment.effect(
    ipd = NULL,
    samples = NULL,
    newpatient = NULL,
    reference = NULL,
    quantiles = c(0.025, 0.5, 0.975)
)

Arguments

ipd IPD object created from running ipdma.model type function
samples MCMC samples found from running ipd.run function
newpatient covariate values of patients that you want to predict treatment effect on. Must have length equal to total number of covariates.
reference group used for finding patient-specific treatment effect. This is only used for deft approach
quantiles quantiles for credible interval of the patient-specific treatment effect

Value
patient-specific treatment effect with credible interval at specified quantiles

References


Examples
```r
ds <- generate_ipdma_example(type = "continuous")
ipd <- with(ds, ipdma.model.onestage(y = y, study = studyid, treat = treat, X = cbind(z1, z2), response = "normal", shrinkage = "none"))

samples <- ipd.run(ipd, pars.save = c("beta", "gamma", "delta"), n.chains = 3, n.burnin = 500, n.iter = 5000)
treatment.effect(ipd, samples, newpatient = c(1,0.5))
```
Index

bipd-package, 2
findMissingPattern, 3
generate_ipdma_example, 4
generate_ipdnma_example, 4
generate_sysmiss_ipdma_example, 5
ipd.run, 6
ipd.run.parallel, 7
ipdma.impute, 8
ipdma.model.deft.onestage, 9
ipdma.model.onestage, 11
ipdnma.model.onestage, 14
treatment.effect, 16