Package 'prome'

February 6, 2026

Title Patient-Reported Outcome Data Analysis with Stan

Version 3.1.0.1

Description Algorithms and subroutines for patient-reported outcome data analysis.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.2

Biarch true

Depends R (>= 3.5.0)

Imports methods, Rcpp (>= 0.12.0), rstan (>= 2.18.1), rstantools (>= 2.1.1), BI

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements GNU make

NeedsCompilation yes

Author Bin Wang [aut, cre] (ORCID: <https://orcid.org/0000-0002-3689-6932>)

Maintainer Bin Wang <bwang831@gmail.com>

Repository CRAN

Date/Publication 2026-02-06 21:20:02 UTC

Contents

`prome-package` *The 'prome' package.*

Description

Algorithms to implenment the Bayesian methods to denoise the measurement errors in patient-reported outcome data with repeated measures. Also, two algorithms are included to discount the subgroup means or proportions for clinical studies with multiple subgroups.

`bate` *Bayesian Hierarchical Model for RPO data with repeated measures*

Description

A Bayesian hierachical model to denoise PRO data using repeated measures.

Usage

```
bate(x0,x1,group,z,x.range,...)
ResponderAnalysis(x,mcid,type="absolute",conf.level=0.95,show=TRUE)
```

Arguments

`x0, x1`	Numeric vector/matrix of observations at T0 (baseline) and T1 (end point) of a study.
`z`	covariates
`group`	group assignments. Current version support one or two groups only
`x.range`	range of data 'x0' and 'x1'
`x`	An R object generated by `memixed`
`mcid`	A threshold to define 'responder'
`type`	The type of responder analysis: absolute or relative changes
`conf.level`	Confidence level of the credible interval
`show`	control whether results should be displayed
`...`	Parameters ("adapt_delta","stepsize","max_treedepth") to improve model fitting/convergence.

Value

- ‘xfit‘: fitted results using stan.
- ‘mu.t0‘: baseline mean.
- ‘sig.t0‘: baseline SD.
- ‘sig.me‘: SD of measurement errors.
- ‘mu.active‘: mean effect size of active treatment.
- ‘sig.active‘: sd of effect size of active treatment.
- ‘mu.sham‘: mean effect size of sham treatment.
- ‘sig.sham‘: sd of effect size of sham treatment.

Examples

```
data(n100x3)
out1  <-  bate(x0=ex100x3$w0,x1=ex100x3$w1,group=ex100x3$group)
out1
ResponderAnalysis(out1,mcid=1,type="abs")
out2  <-  bate(x0=ex100x3$w0,x1=ex100x3$w1,group=ex100x3$group,
    control = list(adapt_delta = 0.8,
               stepsize = 5,
               max_treedepth = 10)
)
out2
ResponderAnalysis(out2,mcid=1,type="abs")
out <- out2
ResponderAnalysis(out,mcid=0.5,type="abs")
ResponderAnalysis(out,mcid=1,type="abs")
ResponderAnalysis(out,mcid=1.5,type="abs")
ResponderAnalysis(out,mcid=0.3,type="relative")
ResponderAnalysis(out,mcid=0.2,type="relative")
ResponderAnalysis(out,mcid=0.1,type="relative")
```

`blinding.BI` *Computing Blinding Index*

Description

To be updated.

Usage

```
blinding.BI(group, guess,iter=100)
```

Arguments

`guess`	guess response from blinding survey
`group`	group assignments. 0='control',1='treatment' or 'active', 2='I don't know', or 'DNK'. Missing values are allowed.
`iter`	imputation iteration.

Value

- blinding index (with bootstrapping CI in the presence of missing responses.

Examples

```
u1      = 5.5 # trt
u2      = 2.0 # ctrl
theta   = 3.2 # sham
sigma2  = 2.5   # v(rij)
ntreat  = 500
nsham   = 500

beta0 = 1.0
beta1 = 2.0
beta2 = 1.0 # no contamination

Tind  = c(rep(1, ntreat), rep(0,nsham))  #treatment group indicator
u1v   = rep(u1,ntreat)
u2v   = rep(u2,nsham)
uv    = c(u1v,u2v)
tauv  = uv - rep(u2, ntreat+nsham)
r = rnorm(ntreat + nsham, mean = 0, sd = sqrt(sigma2))
q = 1/(1 + exp(-(beta0 + beta1*Tind + beta2*(tauv+r))))
bernGen = function(qq){rbinom(1,1,qq)}
I = sapply(q,bernGen)
x = uv + theta*I + r   # fixed sham effect
## I have concerns about the error term(s). x.sham~N(theta,sigma.sham)?
sigma.sham = 1.5
r2 = rnorm(ntreat + nsham, mean = 0, sd = sqrt(sigma.sham))
x = (uv + r) + theta*I #+ r2   # fixed sham effect

out1 <- blinding.BI(group=Tind,guess=I);
out1
```

`blinding.cpe` *Changing point estimator to correct bias due to unblinding in RCTs*

Description

estimate the sham effect and correct effect size

Usage

```
blinding.cpe(x,group,guess,iter=100)
```

Arguments

`x`	outcome variable. numeric vector
`group`	group assignment. Coded as "0"=control and "1"=active/treatment. If 'group' is a factor, the first level will be treated as "control" arm. For example, if there are two values (ie. "ctrl" and "active"), "active" will be treated as the control arm. If the two levels are "control" and "treatment", "control" will be treated as the control arm.
`guess`	responses to the blinding survey question. The response corresponding to positive sham effect needs to be coded as "1", and the rest as "0". If the possible responses are "Active", "Control" and "I don't know", code "guess" as "1" for "Active" and "0" for the others if a subject responded "active" is expect to have positive sham effect. Otherwise, if a subject responded "active" is expect to have negative sham effect, code "guess" as "0" for "Active" and "1" for the others.
`iter`	imputation iteration.

Details

To be added

Value

3 estimates, BI indices.

Examples

```
u1      = 5.5 # trt
u2      = 2.0 # ctrl
theta   = 3.2 # sham
sigma2  = 2.5   # v(rij)
ntreat  = 500
nsham   = 500

beta0 = 1.0
beta1 = 2.0
beta2 = 1.0 # no contamination

Tind  = c(rep(1, ntreat), rep(0,nsham))  #treatment group indicator
u1v   = rep(u1,ntreat)
u2v   = rep(u2,nsham)
uv    = c(u1v,u2v)
tauv  = uv - rep(u2, ntreat+nsham)
r = rnorm(ntreat + nsham, mean = 0, sd = sqrt(sigma2))
q = 1/(1 + exp(-(beta0 + beta1*Tind + beta2*(tauv+r))))
bernGen = function(qq){rbinom(1,1,qq)}
I = sapply(q,bernGen)
x = uv + theta*I + r   # fixed sham effect
```

```
## I have concerns about the error term(s). x.sham~N(theta,sigma.sham)?
sigma.sham = 1.5
r2 = rnorm(ntreat + nsham, mean = 0, sd = sqrt(sigma.sham))
x = (uv + r) + theta*I #+ r2   # fixed sham effect

out1 <- blinding.cpe(x=x,group=Tind,guess=I);
out1

##data(bd012)
##blinding.cpe(x=bd012$y, group=bd012$group,guess=bd012$guess)
##data(bd011)
##blinding.cpe(x=bd011$y, group=bd011$group,guess=bd011$guess)
##data(bd010)
##blinding.cpe(x=bd010$y, group=bd010$group,guess=bd010$guess)
```

`blinding.test` *Latent Shift Logistic Regression*

Description

To be updated.

Usage

```
blinding.test(x, group, guess, mu0 = 0, s0 = 1,...)
```

Arguments

`x, guess`	outcome variable and guess response from blinding survey
`group`	group assignments. Current version support one or two groups only
`mu0, s0`	initial mean and sd of the latent variable of having sham effects
`...`	Parameters ("adapt_delta","stepsize","max_treedepth") to improve model fitting/convergence.

Value

- ‘sig.sham‘: sd of effect size of sham treatment.

Examples

```
u1      = 5.5 # trt
u2      = 2.0 # ctrl
theta   = 3.2 # sham
sigma2  = 2.5   # v(rij)
ntreat  = 500
nsham   = 500

beta0 = 1.0
```

```
beta1 = 2.0
beta2 = 1.0 # no contamination

Tind  = c(rep(1, ntreat), rep(0,nsham))  #treatment group indicator
u1v   = rep(u1,ntreat)
u2v   = rep(u2,nsham)
uv    = c(u1v,u2v)
tauv  = uv - rep(u2, ntreat+nsham)
r = rnorm(ntreat + nsham, mean = 0, sd = sqrt(sigma2))
q = 1/(1 + exp(-(beta0 + beta1*Tind + beta2*(tauv+r))))
bernGen = function(qq){rbinom(1,1,qq)}
I = sapply(q,bernGen)
x = uv + theta*I + r   # fixed sham effect
## I have concerns about the error term(s). x.sham~N(theta,sigma.sham)?
sigma.sham = 1.5
r2 = rnorm(ntreat + nsham, mean = 0, sd = sqrt(sigma.sham))
x = (uv + r) + theta*I #+ r2   # fixed sham effect

out1 <- blinding.test(x=x,group=Tind,guess=I);
out1

##data(bd012)
##blinding.test(x=bd012$y, group=bd012$group,guess=bd012$guess)
##data(bd011)
##blinding.test(x=bd011$y, group=bd011$group,guess=bd011$guess)
##data(bd010)
##blinding.test(x=bd010$y, group=bd010$group,guess=bd010$guess)
```

`blindingdata` *Simulated blinding data*

Description

Simulated blinding survey datasets .

Format

Key variables

`y`	vector	outcome variable
`guess`	vector	guess of treatments
`group`	vector	group assignment

Examples

```
data(bd012)
names(bd012)
```

ex100x3 *Sample PRO Data With Repeated Measures*

Description

A simulated data set of patient-reported outcomes with repeated measures.

Format

A data frame with observations at beaseline and at a follow-up time.

`w0`	matrix	measures at baseline
`w1`	matrix	measures at follow-up time
`group`	character	group assignment

MeanHM *Bayesian Hierarchical Model for Information Borrowing for Means*

Description

To compute the mean values of subgroups based on a Bayesian hierarchical model.

Usage

```
MeanHM(x,sigma)
```

Arguments

`x`	Numeric vector of observations for the subgroups.
`sigma`	hyper-parameter. to be estimated or can be given.

Value

- ‘theta‘: population mean.
- ‘sigma‘: population standard deviation.

Examples

```
x1 <- rnorm(100,2,1)
x2 <- rnorm(100,3,1.5)
x3 <- rnorm(100,4,1.9)
x <- cbind(x1,x2,x3)
MeanHM(x,sigma=0.5)
```

`PropHM` *Bayesian Hierarchical Model for Information Borrowing for Proportions*

Description

To compute the proportions of the subgroups assuming the subgroups follow the same binomial distribution with parameter `p`. The approach on partial pooling by Bob Carpenter has been used – "Hierarchical Partial Pooling for Repeated Binary Trials" https://mc-stan.org/users/documentation/case-studies/pool-binary-trials.html

Usage

```
PropHM(x, n, kappa)
```

Arguments

`x`	Numeric vector of events.
`n`	Numberic vector of group sample sizes.
`kappa`	`kappa=alpha+beta>1`. Must be given if the number of subgroups is 2.

Value

- ‘data‘: data with estimates.
- ‘alpha‘: parameter of the beta distribution.
- ‘beta‘: parameter of the beta distribution.

Examples

```
out <- PropHM(x=c(5,10,2),n=c(20,50,30))
```

`xover` *Bayesian analysis of 2x2 crossover trial data*

Description

A Bayesian hierachical model to analysis data from 2x2 (AB/BA) crossover trials.

Usage

```
  xover(group,y1,y2,y0,...)
```

Arguments

`y0, y1, y2`	vectors of data from baseline, period 1, and period 2, respectively.
`group`	group or treatment sequence.
`...`	other parameters, i.e. 'control' for model fitting.

Value

- ‘stat‘: summary statistics.
- ‘best‘: estimates using Bayesian analysis.

Examples

```
xover(y0=rnorm(20,34,1.5),y1=rnorm(20,30,2),
      y2=rnorm(20,25,1.5),group=round(runif(20)<0.5))
```

Index