Package: BayesRS (via r-universe)

October 15, 2026

Type Package

Title Bayes Factors for Hierarchical Linear Models with Continuous Predictors

Version 0.1.3

Depends R (>= 3.0.0)

Imports rjags, ggplot2, metRology, grid, reshape, methods, coda

Suggests R.rsp, testthat

VignetteBuilder R.rsp

Description Runs hierarchical linear Bayesian models. Samples from the posterior distributions of model parameters in JAGS (Just Another Gibbs Sampler; Plummer, 2017, <http://mcmc-jags.sourceforge.net>). Computes Bayes factors for group parameters of interest with the Savage-Dickey density ratio (Wetzels, Raaijmakers, Jakab, Wagenmakers, 2009, <doi:10.3758/PBR.16.4.752>).

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Author Mirko Thalmann [aut, cre], Marcel Niklaus [aut], Klaus Oberauer [ths], John Kruschke [ctb]

Maintainer Mirko Thalmann <mirkothalmann@hotmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2018-04-06 05:39:35 UTC

RemoteUrl https://github.com/cran/BayesRS

RemoteRef HEAD

RemoteSha c839a7bbe57f260123f5e8aed449b67ed3f5cd08

Contents

bayesrsdata 2
modelrun 3

Index **8**

`bayesrsdata` *Example Data Set*

Description

Example data set used for showing functionality of `modelrun`. The examples give the code used for simulating the data set.

Usage

```
bayesrsdata
```

Format

A data.frame with 1200 rows and 4 variables

Examples

```
## Not run:
require(MASS)

nsubj <- 40             # number of participants
nobs <- 30              # number of observations per cell
ncont <- 1              # number of continuous IVs
ncat <- 1               # number of categorical IVs
ntrials <- nobs * ncont * ncat #total number of trials per subject
xcont <- seq(1,5,1)     # values of continuous IV
xcont.mc <- xcont-mean(xcont) # mean-centered values of continuous IV
xcat <- c(-.5,.5)             # Simple coded categorical IV
eff.size.cont <- c(0.3)       # effect size of continuous IV
eff.size.cat <- c(0.8)       # effect size of categorical IV
eff.size.interaction <- c(0)  # effect size of interaction
correlation.predictors <- 0.5   # correlation between b<-subject slopes of the two predictors
intercept <- 0          # grand intercept
error.sd <- 1           # standard deviation of error term

#-------------------------
#  Create Simulated Data -
#-------------------------
#  correlation between by-subject continuous slope, and by-subject categorical slope
subj.cont1.cat1.corr <- mvrnorm(n = nsubj,
                                mu = c(eff.size.cont,eff.size.cat),
                                Sigma = matrix(data = c(1,correlation.predictors,
                                                        correlation.predictors,1),
```

```
                                                  nrow = 2, ncol = 2, byrow = TRUE),
                                  empirical = TRUE)

b.cont.subj <- data.frame(subject = 1:nsubj, vals = subj.cont1.cat1.corr[,1])
b.cat.subj <- data.frame(subject = 1:nsubj, vals = subj.cont1.cat1.corr[,2])
b.subj.rand <- data.frame(subject = 1:nsubj, vals = rnorm(n = nsubj, mean = 0, sd = 1))
b.ia.subj <- data.frame(subject = 1:nsubj, vals = rnorm(n = nsubj,
                                                  mean = eff.size.interaction, sd = 1))

# generate according to lin reg formula
bayesrsdata <- data.frame(subject = rep(1:nsubj, each = ntrials),
                          x.time = rep(xcont, each = ntrials/5),
                          x.domain= rep(xcat, each = ntrials/10))

bayesrsdata$y <- 0

for (i in 1:nrow(bayesrsdata)){
  bayesrsdata$y[i] <- b.subj.rand$vals[bayesrsdata$subject[i]==b.subj.rand$subject] +
    bayesrsdata$x.time[i] * (eff.size.cont+b.cont.subj$vals[bayesrsdata$subject[i]==
                                                        b.cont.subj$subject]) +
    bayesrsdata$x.domain[i] * (eff.size.cat+b.cat.subj$vals[bayesrsdata$subject[i]==
                                                        b.cat.subj$subject]) +
    bayesrsdata$x.time[i] * bayesrsdata$x.domain[i] *
    (eff.size.interaction+b.ia.subj$vals[bayesrsdata$subj[i]==b.ia.subj$subject])
}

# add measurement error
bayesrsdata$y <- bayesrsdata$y + rnorm(n = nrow(bayesrsdata), mean = 0, sd = 1)

# create final data set
recvars <- which(names(bayesrsdata) %in% c("subject", "item", "x.domain"))
bayesrsdata[,recvars] <- lapply(bayesrsdata[,recvars], as.factor)

save(bayesrsdata, file= "bayesrsdata.rda")


## End(Not run)
```

`modelrun` *Bayes Factors, Posterior Samples, & DIC*

Description

Computes Bayes Factors for hierarchical linear models including continuous predictors using the Savage-Dickey density ratio

Usage

```
modelrun(data, dv, dat.str, randvar.ia = NULL, corstr = NULL,
  nadapt = NULL, nburn = NULL, nsteps = NULL, checkconv = NULL,
  mcmc.save.indiv = NULL, plot.post = NULL, dic = NULL, path = NULL)
```

Arguments

data	a data.frame object with the data to be fitted in the long format.
dv	string indicating the dependent variable of the model. Has to be normally distributed.
dat.str	a data.frame object indicating the hierarchical structure in the model with column names "iv" and "type" and an arbitrary number of random variables as the following column names. iv indicates the name of an independent variable as in data, type its scale of measurement ("cont" for continuous or "cat" for categorical), and the following entries indicate whether a random effect should be modeled for this variable (1) or not (0). Continuous variables have to be entered before categorical variables. The name for the random variable(s) has to be the same as in data. A categorical variable with n levels is entered as n - 1 simple codes into the model with the first level of the variable being the reference category.
randvar.ia	a list containing n matrix objects with n being the number of random variables. In each matrix the lower triangle can be used to declare the respective two-way interaction as random within the specific random variable. The row- and column- ordering of independent variables is the same as in dat.str. When not specified, interactions are only modeled as fixed effects by default.
corstr	a list containing n matrix objects with n being the number of random variables. In each matrix the lower triangle can be used to assign correlations between predictors (including the intercept) for each random effect. The first row and column in each matrix object represents the intercept. The following rows and columns represent the independent variables with the same ordering as in dat.str. When not specified, no correlations are modeled by default.
nadapt	number of MCMC steps to adapt the sampler (2000 by default).
nburn	number of MCMC steps to burn in the sampler (2000 by default).
nsteps	number of saved MCMC steps in all chains (100'000 by default).
checkconv	indicates that convergence statistics of the main model parameters should be returned in the console and that figures of the chains should be plotted when set to 1 (0 by default).
mcmc.save.indiv	indicates that the chains should be saved in a data.frame object when set to 1 (0 by default).
plot.post	indicates that the 95 percent highest-density interval of the posterior of the group parameters should be plotted as a figure with the corresponding Bayes Factors when set to 1 (0 by default).
dic	indicates that the deviation information criterion (Spiegelhalter, Best, Carlin, & Linde, 2002) should be computed for a given model when set to 1 (0 by default).
path	defines directory where model is saved as .txt file and model name. Is set to file.path(tempdir(), "model.txt") by default.

Details

The argument `corstr` can be used to model correlations between (a) pairs of predictors and (b) more than two predictors. When both is done within the same random variable, a predictor can only appear in (a) or (b).

`modelrun` z-standardizes the dependent variable and the continuous independent variables. To obtain the posteriors in the original scale they have to be retransformed.

Savage Dickey: Bayes Factors are computed with the Savage-Dickey density ratio. We use the normal approximation (e.g., Wetzels, Raaijmakers, Jakab, & Wagenmakers, 2009) to estimate the density of the posterior.

Value

returns a list with components:

- `bf`: a `data.frame` object with the Bayes Factor estimates of the group parameters (aka fixed effects).
- `mcmcdf`: a `data.frame` object with the saved MCMC chains.
- `dic`: DIC of the fitted model.

Author(s)

Thalmann, M., Niklaus, M. Part of this package uses code from John Kruschke.

References

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & van der Linde, A. (2002). Bayesian measures of model complexity and fit. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 64(4), 583.

Wetzels, R., Raaijmakers, J. G. W., Jakab, E., & Wagenmakers, E.-J. (2009). How to quantify support for and against the null hypothesis: A flexible WinBUGS implementation of a default Bayesian t test. Psychonomic Bulletin & Review, 16(4), 752-760. https://doi.org/10.3758/PBR.16.4.752

Examples

```
data(bayesrsdata) #load data
```

```
## ------------------------------------------------------------------
## Example 1: Estimation of Bayes Factors from a continuous
## independent variable (IV) with random slopes
## - repeated measures for each participant
## - continuous variable with 5 values: x.time
## ------------------------------------------------------------------

## JAGS Sampler Settings
```

```
# -----------------
# nr of adaptation, burn-in, and saved mcmc steps only for exemplary use
nadapt = 2000           # number of adaptation steps
nburn = 2000            # number of burn-in samples
mcmcstep = 100000       # number of saved mcmc samples, min. should be 100'000

# Define model structure;
dat.str <- data.frame(iv = c("x.time"),
                      type = c("cont"),
                      subject = c(1))
# name of random variable (here 'subject') needs to match data frame

# Run modelrun function
out <- modelrun(data = bayesrsdata,
                dv = "y",
                dat.str = dat.str,
                nadapt = nadapt,
                nburn = nburn,
                nsteps = mcmcstep,
                checkconv = 0)

# Obtain Bayes factor
bf <- out[[1]]
bf

## ------------------------------------------------------------------
## Example 2: Estimation of Bayes Factors from a continuous
## independent variable with random slopes that
## are correlated with the random slopes of a categorical variable.
## - Repeated measures for each participant
## - a continuous IV with 5 values: x.time
## - a categorical variable with 2 levels: x.domain
## ------------------------------------------------------------------

## JAGS Sampler Settings
# nr of adaptation, burn-in, and saved mcmc steps only for exemplary use
# -----------------
nadapt = 2000         # number of adaptation steps
nburn = 2000           # number of burn-in samples
mcmcstep = 100000       # number of saved mcmc samples, min. should be 100'000

# Define model structure;
# order of IVs: continuous variable(s) needs to go first
dat.str <- data.frame(iv = c("x.time", "x.domain"),
                      type = c("cont", "cat"),
                      subject = c(1,1))
# name of random variable (here 'subject') needs to match data frame

# Define random effect structure on interaction for each random variable
ias.subject <- matrix(0, nrow=nrow(dat.str), ncol = nrow(dat.str))
ias.subject[c(2)] <- 1
randvar.ia <- list(ias.subject)
```

```
# Define correlation structure between predictors within a random variable
cor.subject <- matrix(0, nrow=nrow(dat.str)+1, ncol = nrow(dat.str)+1)
cor.subject[c(2,3,6)] <- 1
corstr <- list(cor.subject)

# Run modelrun function
out <- modelrun(data = bayesrsdata,
                dv = "y",
                dat.str = dat.str,
                randvar.ia = randvar.ia,
                nadapt = nadapt,
                nburn = nburn,
                nsteps = mcmcstep,
                checkconv = 0,
                mcmc.save.indiv = 1,
                corstr = corstr)

# Obtain Bayes factors for continous main effect,
# categorical main effect, and their interaction
bf <- out[[1]]
bf
```

Index

∗ **dataset**
bayesrsdata, 2

bayesrsdata, 2

modelrun, *2*, 3