Package: robustBLME (via r-universe)

September 7, 2024

Type Package

Title Robust Bayesian Linear Mixed-Effects Models using ABC

Version 0.1.2

Date 2017-06-01

Maintainer Erlis Ruli `<erlisr@yahoo.it>`

Description Bayesian robust fitting of linear mixed effects models though weighted likelihood equations and approximate Bayesian computation.

Depends R (>= 3.0.0), lme4 (>= 1.1.12)

Suggests devtools

License GPL-2

URL `https://github.com/erlisR/robustBLME`

BugReports `https://github.com/erlisR/robustBLME/issues`

LazyData TRUE

Imports Rcpp (>= 0.12.1), numDeriv, mvtnorm, utils, parallel, doParallel, foreach, iterators, stats

LinkingTo Rcpp, RcppArmadillo

Author Erlis Ruli [aut, cre], Nicola Sartori [aut], Laura Ventura [aut]

RoxygenNote 6.0.1

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

RemoteUrl https://github.com/erlisr/robustblme

RemoteRef HEAD

RemoteSha 695e75f591f8d0b9445be7011b2f3153b5e16ea0

Contents

`ergoStool` *Ergometrics experiment with stool types*

Description

The `ergoStool` data frame has 36 rows and 3 columns.

Format

This data frame contains the following columns:

effort a numeric vector giving the effort (Borg scale) required to arise from a stool.

Type a factor with levels `T1`, `T2`, `T3`, and `T4` giving the stool type.

Subject an ordered factor giving a unique identifier for the subject in the experiment.

Details

Devore (2000) cites data from an article in *Ergometrics* (1993, pp. 519-535) on “The Effects of a Pneumatic Stool and a One-Legged Stool on Lower Limb Joint Load and Muscular Activity.”

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.9)

Devore, J. L. (2000), *Probability and Statistics for Engineering and the Sciences (5th ed)*, Duxbury, Boston, MA.

`hpd` *Empirical Highest Posterior Density Interval*

Description

Computes the Highest Posterior Density (HPD) interval from a posterior sample. Works only for scalar marginal posteriors.

Usage

```
hpd(x, prob = 0.95)
```

Arguments

`x`	a univariate or a posterior sample for a scalar parameter.
`prob`	posterior probability content.

Value

vector of two doubles.

`kdeFSBT` *Full Significance Bayesian Testing*

Description

Performs Full Significance Bayesian Testing (FSBT) for univariate sharp null hypothesis based on a posterior sample. The marginal posterior density is obtained by kernel density estimation from `sim.sample`.

Usage

```
kdeFSBT(H0, sim.sample)
```

Arguments

`H0`	a scalar value under the null hypothesis.
`sim.sample`	a sample from the marginal posterior distribution.

Value

double

References

Pereira, C. A. d. B., Stern, J. M. and Wechsler, S. (2008) Can a significance test be genuinely Bayesian? *Bayesian Analysis* **3**, 79-100.

Examples

```
x <-  rnorm(1000, 0, 1)
kdeFSBT(-1, x)
```

`rblme` *Fits robust Bayesian Linear Mixed-effects Models (LMM) to data via robust REML estimating functions.*

Description

This function fits robust Bayesian LMMs to data via robust REML estimating functions. The latters are those proposed by Richardson & Welsh (1995), which are robustified versions of restricted maximum likelihood (REML) estimating equations. Posterior sampling is done with an ABC-MCMC algorithm, where the data are summarised through a rescaled version of the aforementioned estimating functions; see Ruli et al. (2017) for the properties and details of the method. The current package version (0.1.2) supports only models with a single random effects. Extensions to more general settings will be provided in the future versions of the package.

Usage

```
rblme(nabc, h.obj, chain.control = list(trace.init = NULL, thin.by = NULL),
        n.cores = 1)
```

Arguments

`nabc`	number of posterior samples.
`h.obj`	list of objects as returned by the `tune.h` function. Hence `tune.h` must be called first.
`chain.control`	parameters that control the tracing and the thinning of the chain(s).
`n.cores`	number of cores for parallel computation on non Windows machines. For `n.cores`>2, `n.cores` chains are run each on a different core with using the same parameters but with a different random seed.

Value

list or list of lists with elements `abc` and `effi`. In case of `n.cores`=1, `effi` is the actual acceptance rate of the ABC-MCMC algorithm whereas in `abc` are stored the posterior samples. The latters are stored as a $(q+c)\times$nabc matrix, where q is the number of fixed effects, i.e. the number of columns in the design matrix and $c = 2$ is the number of variance components. Hence, the first q rows of the matrix `abc` give the posterior samples for the fixed effects and the last two rows give the posterior samples for the log-variances of the fixed effects and the residual term, respectively. If `n.cores` > 1, i.e. if simulations are performed in parallel, then a list of lists is returned, where each element of the list is a list with elements `abc` and `effi`, where `abc` and `effi` are as those aforementioned.

References

Ruli E., Sartori N. & Ventura L. (2017) Robust approximate Bayesian inference with an application to linear mixed models. `https://arxiv.org/abs/1706.01752`

Richardson A. M. & Welsh A. H. (1995) Robust restricted maximum likelihood in mixed linear models. *Biometrics* **51**, 1429-1439.

See Also

`tune.h`, `ergoStool`.

Examples

```
## The following example is meant for function documentation.
## For realistic use probably you'll need to take a larger sample and choose a
## "better" bandwidth h.

data(ergoStool)

require(lme4)
fm1 <- lmer(effort~Type + (1| Subject), data = ergoStool)

## tune h to get 0.8% acceptance
hopt <- tune.h(effort~Type + (1|Subject), data = ergoStool, n.samp = 1e+4,
               acc.rate = 0.01, n.sim.HJ = 100, grid.h = seq(0.3, 0.7, len = 3),
               prior = list(beta.sd = 10, s2.scale = 5), n.cores = 1)

## draw posterior samples with hopt.
abc.tmp <- rblme(nabc = 1e+4, h.obj = hopt,
                 n.cores = 1)

# process ABC samples
abc.sim <- t(abc.tmp$abc)
abc.sim[,c(5,6)] <- exp(abc.sim[,c(5,6)])

# ABC posterior
colMeans(abc.sim)

# REML estimates
summary(fm1)
```

`tune.h` *Tune ABC distance bandwidth*

Description

Tunes the bandwidth h of the ABC distance to get the desired level of acceptance rate specified via `acc.rate`. Besides tuning h, the function also builds the relevant quantities needed for running `rblme`. For generating such quantities an internal call to `lmer` is performed.

Usage

```
tune.h(formula, data, ..., n.samp = 1e+5, n.sim.HJ = 500, acc.rate, grid.h, prior,
       cHub = 1.345, cHub2 = 2.07,
       init, n.cores = 1, use.h)
```

Arguments

`formula`	two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a ~ operator and the terms, separated by + operators on the right. The `"|"` character separates an expression for a model matrix and a grouping factor.
`data`	optional data frame containing the variables named in `formula`. By default the variables are taken from the environment of `lmer` called internally.
`...`	other arguments to be passed to `lmer`. Currently none is used.
`n.samp`	number of pilot posterior samples to be drawn with ABC for each value of `grid.h`.
`n.sim.HJ`	number of simulations to be used for computing the sensitivity and variability matrices.
`acc.rate`	desired acceptance rate of the ABC-MCMC algorithm.
`grid.h`	grid of h values within which the "optimal" value is to be found.
`prior`	named list of user-defined prior hyper-parameters. See "Details" below.
`cHub`	tuning constant of the Huber function for the location parameter.
`cHub2`	tuning constant of the Huber proposal 2 function for the scale parameter.
`init`	optional object to use for starting values. Currently ignored as initial values are taken from `lmer`.
`n.cores`	number of cores for parallel computation on non Windows machines.
`use.h`	bandwidth to be used for the ABC distance. If provided, no tuning for h is performed and `acc.rate` is ignored.

Details

Given a specification of the `formula` and `data` the function calls internally `rlmer` and extracts from the resulting object all the necessary quantities. Then proceeds by finding the solution of the REML II robust estimating equations (Richardson & Welsh 1995), with the REML estimate used as starting point. The sensitivity and the variability matrices are computed by simulation at the solution of the robust REML II estimating equation. Depending on whether `use.h` or `acc.rate` and `grid.h` are specified, the function has a different behaviour. If `acc.rate` and `grid.h` are provided, then an adaptive step is performed in order to get an "optimal" h which gives the desired acceptance rate `acc.rate`. IN particular, for each value of `grid.h`, the function draws `n.samp` posterior samples with the ABC-MCMC algorithm and saves the resulting acceptance rate. Lastly, a function is built via a smoothing spline with acceptance rates being the xs and `grid.h` being the ys. The "optimal" value of h is found, within `grid.h`, as the prediction the spline function at `acc.rate`. If you already have an h value in mind then specify it via `use.h` and leave `grid.h` and `acc.rate` unspecified. Note that, in this case the acceptance rate of the ABC-MCMC algorithm may not be the one you wish to obtain since it depends in some complicated way also from $use.h$. Currently, the prior for the q fixed effects is the product of q scalar normals with mean zero and user-specified variance `beta.sd` (see Examples) equal for all the parameters. For the variance components the prior is a halfCauchy with user-specified scale `s2.scale`. Both variance parameters are assumed to have equal prior scale.

Value

list.

References

Ruli E., Sartori N. & Ventura L. (2017) Robust approximate Bayesian inference with an application to linear mixed models. `https://arxiv.org/abs/1706.01752`

Richardson A. M. & Welsh A. H. (1995) Robust restricted maximum likelihood in mixed linear models. *Biometrics* **51**, 1429-1439.

See Also

`rblme`, `ergoStool`.

Index