Package 'yodel'

July 21, 2025

Title A General Bayesian Model Averaging Helper

Version 1.0.0

Description Provides helper functions to perform Bayesian model averaging using Markov chain Monte Carlo samples from separate models. Calculates weights and obtains draws from the model-averaged posterior for quantities of interest specified by the user. Weight calculations can be done using marginal likelihoods or log-predictive likelihoods as in Ando, T., & Tsay, R. (2010) <doi:10.1016/j.ijforecast.2009.08.001>.

License MIT + file LICENSE

URL https://github.com/rich-payne/yodel

Imports dplyr (>= 1.0), purrr (>= 0.3), rlang (>= 0.4)

Encoding UTF-8

RoxygenNote 7.2.3

Suggests testthat

NeedsCompilation no

Author Richard Payne [aut, cre],
Eli Lilly and Company [cph]

Maintainer Richard Payne <paynestatistics@gmail.com>

Repository CRAN

Date/Publication 2024-04-16 08:40:02 UTC

Contents

bma *Posterior Weights and Model Averaging Setup*

Description

Calculate posterior weights of each model and optionally supply MCMC samples and functions (through the `bma_model()` function) to calculate a quantity of interest from each model using the `posterior()` function.

Usage

```
bma(..., seed = sample(.Machine$integer.max, 1))

model_bma_predictive(
  log_post_pred,
  adjustment = 0,
  w_prior = 1,
  mcmc = NULL,
  fun = NULL
)

model_bma_marginal(log_marginal, w_prior = 1, mcmc = NULL, fun = NULL)
```

Arguments

`...`	Named calls to the `bma_model()` function.
`seed`	an integer which is used to specify the seed when sampling from the different models (e.g. in `posterior()`).
`log_post_pred`	a matrix containing the log likelihood for each observation on each iteration of the MCMC. The matrix should have dimensions (number-of-MCMC-iteration) by (number of observations).
`adjustment`	an adjustment to be applied to the posterior log-predictive likelihood. A simple bias correction in Ando & Tsay (2010) is: - (number of parameters in the model) / 2.
`w_prior`	the prior weight for the model.
`mcmc`	a named list (or dataframe) of MCMC samples of model parameters.
`fun`	a function which takes the MCMC samples and returns a value of interest.
`log_marginal`	The log marginal likelihood of the model.

Details

It is required that if MCMC samples are supplied, that each MCMC run must have the same number of collected samples.

Value

bma: A list containing the prior and posterior weights for each model, the sampled model (`model_index`) at each MCMC iteration and the arguments supplied to each `bma_model()` call.

model_bma: A named list of the arguments, with a "yodel_bma_candidate" class attached.

model_bma: A named list of the arguments, with a "yodel_bma_candidate" class attached.

References

Ando, T., & Tsay, R. (2010). Predictive likelihood for Bayesian model selection and averaging. International Journal of Forecasting, 26(4), 744-763.

Examples

```
# Minimal example
fit <- bma(
  linear = model_bma_predictive(
    # mcmc = data.frame(b1 = 1:5, b2 = 11:15, sigma = seq(.1, .5, .1)),
    log_post_pred = matrix(log(1:100), 5, 20),
    adjustment = - 3 / 2,
    w_prior = .5
  ),
  quad = model_bma_predictive(
    # mcmc = data.frame(b1 = 1:5 / 2, b2 = 11:15 / 2, b3 = 5:1, sigma = seq(.1, .5, .1)),
    log_post_pred = matrix(log(2:101), 5, 20),
    adjustment = - 4 / 2,
    w_prior = .5
  )
)

fit$w_prior
fit$w_post
```

`posterior` *Calculate Posterior Quantities*

Description

Calculate posterior quantities specifically of interest for a given model.

Usage

```
posterior(x, ...)
```

Arguments

`x`	MCMC output.
`...`	additional arguments passed to S3 methods.

Value

a dataframe or tibble with the posterior probabilities.

Examples

```
# functions which caclulate the dose response for a linear and quadratic model
fun_linear <- function(x, dose) {
  mean_response <- x$b1 + x$b2 * dose
  data.frame(iter = 1:nrow(x), dose = dose, mean = mean_response)
}
fun_quad <- function(x, dose) {
  mean_response <- x$b1 + x$b2 * dose + x$b3 * dose ^ 2
  data.frame(iter = 1:nrow(x), dose = dose, mean = mean_response)
}

# Bayesian model averaging
fit <- bma(
  linear = model_bma_predictive(
    mcmc = data.frame(b1 = 1:5, b2 = 11:15, sigma = seq(.1, .5, .1)),
    log_post_pred = matrix(log(1:100), 5, 20),
    adjustment = - 3 / 2,
    w_prior = .5,
    fun = fun_linear
  ),
  quad = model_bma_predictive(
    mcmc = data.frame(b1 = 1:5 / 2, b2 = 11:15 / 2, b3 = 5:1, sigma = seq(.1, .5, .1)),
    log_post_pred = matrix(log(2:101), 5, 20),
    adjustment = - 4 / 2,
    w_prior = .5,
    fun = fun_quad
  )
)

# posterior samples using Bayesian model averaging
posterior(fit, dose = 1)
posterior(fit, dose = 2)
```

---

posterior.yodel_bma *Posterior Samples from Bayesian Model Averaging*

Description

Calculate posterior quantities of interest using Bayesian model averaging.

Usage

```
## S3 method for class 'yodel_bma'
posterior(x, ...)
```

Arguments

x	output from a call to bma().
...	additional arguments to be passed to each of the functions used to calculate the quantity of interest.

Value

A dataframe with the posterior samples for each iteration of the MCMC. The dataframe will have, at a minimum, the columns "iter" and "model" indicating the MCMC iteration and the model that was used in the calculations. The functions used for each model are defined within the `model_bma()` function and used in the `bma()` function. See the example below.

Examples

```
# functions which caclulate the dose response for a linear and quadratic model
fun_linear <- function(x, dose) {
  mean_response <- x$b1 + x$b2 * dose
  data.frame(iter = 1:nrow(x), dose = dose, mean = mean_response)
}
fun_quad <- function(x, dose) {
  mean_response <- x$b1 + x$b2 * dose + x$b3 * dose ^ 2
  data.frame(iter = 1:nrow(x), dose = dose, mean = mean_response)
}

# Bayesian model averaging
fit <- bma(
  linear = model_bma_predictive(
    mcmc = data.frame(b1 = 1:5, b2 = 11:15, sigma = seq(.1, .5, .1)),
    log_post_pred = matrix(log(1:100), 5, 20),
    adjustment = - 3 / 2,
    w_prior = .5,
    fun = fun_linear
  ),
  quad = model_bma_predictive(
    mcmc = data.frame(b1 = 1:5 / 2, b2 = 11:15 / 2, b3 = 5:1, sigma = seq(.1, .5, .1)),
    log_post_pred = matrix(log(2:101), 5, 20),
    adjustment = - 4 / 2,
    w_prior = .5,
    fun = fun_quad
  )
)

# posterior samples using Bayesian model averaging
posterior(fit, dose = 1)
posterior(fit, dose = 2)
```

Index