Introduction to Bayesian Marginal Additive Models

2023-03-29

Contents

In this document, we illustrate the R package `bmam` for Bayesian marginal additive models (BMAM).

```r
# install.packages('devtools')
# devtools::install_github('tianyi-pan/BMAM')
library(bmam)
```

Basic example of Bayesian marginal additive models

We use the function `bmam::GenBinaryY` to generate datasets, to illustrate the main features of `bmam` package. We consider the binary case, where the outcome variable Y follows a Bernoulli distribution. The link function $g(\cdot)$ is set as the logit function.

The cluster-level covariate x_{ij1} and the two subject-level covariates x_{ij2} and x_{ij3} were generate from $\mathrm{Unif}(-1, 1)$. We set the correlation within cluster as 0.5 for x_{ij2} i.e., $\mathrm{corr}(x_{ij2}, x_{ij'2}) = 0.5, j \neq j'$, and the linear term x_{ij3} independent among all observations. We generate Y_{ij} according to the marginal model,

$$\mathrm{logit}\left\{E[Y_{ij}|\boldsymbol{x}_{ij}]\right\} = f_1^{\mathrm{M}}(x_{ij1}) + f_2^{\mathrm{M}}(x_{ij2}) + \beta_3^{\mathrm{M}} x_{ij3},$$

The underlying conditional model is specified as,

$$\mathrm{logit}\left\{E[Y_{ij}|\boldsymbol{x}_{ij}, \boldsymbol{u}_i]\right\} = f_1^{\mathrm{C}}(x_{ij1}) + f_2^{\mathrm{C}}(x_{ij2}) + \beta_3^{\mathrm{C}} x_{ij3} + u_{i0} + u_{i1} x_{ij3},$$

with a random intercept and slopes dependence structure,

$$\begin{bmatrix} u_{i0} \\ u_{i1} \end{bmatrix} \sim \mathrm{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_0^2 & \rho\sigma_0\sigma_1 \\ \rho\sigma_0\sigma_1 & \sigma_1^2 \end{bmatrix} \right),$$

where $\sigma_0 = 2, \sigma_1 = 1$ and $\rho = 0.5$.

We specified $\beta_3^{\mathrm{C}} = \beta_3^{\mathrm{M}} = 1$, and the smooth associations as

$$\begin{aligned} f_1^{\mathrm{C}}(x) = f_1^{\mathrm{M}}(x) =& \frac{3}{2}\sin(\pi x) - 2x, \\ f_2^{\mathrm{C}}(x) = f_2^{\mathrm{M}}(x) =& \frac{3}{2}\cos\left(2\pi x + \frac{\pi}{2}\right). \end{aligned}$$

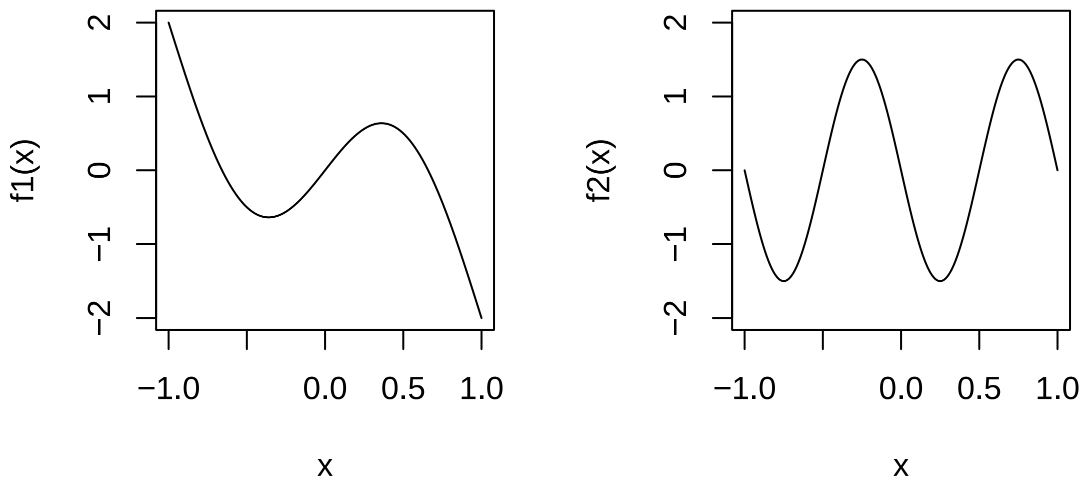
We generate a dataset with $K = 300$ clusters and $N_k = 10$ units in each cluster.

```r
set.seed(4321)

## smooth function setting
f1temp <- function(x){(1.5*sin(pi*x)-2*x)}
f1 <- function(x) f1temp(x)-f1temp(0)

f2temp <- function(x){(1.5*cos(2*pi*(x+1/4)))}
f2 <- function(x) f2temp(x)-f2temp(0)

beta3 <- 1

K <- 300
Nk <- 10
x1 <- round(runif(K*Nk,-1,1),3)
x2 <- round(runif(K*Nk,-1,1),3)
x3 <- round(runif(K*Nk,-1,1),3)


fx1 <- f1(x1)
fx2 <- f2(x2)
beta <- c(0,1,1,beta3)
trueSigma <- matrix(c(4,1,1,1),nrow=2,ncol=2)


nclust <- rep(Nk,K) # number of units in each cluster
id <- rep(seq(K), nclust) # id of each unit
data  <- data.frame(id, x1, fx1, x2, fx2, x3) # generate data (without y)
data  <- data[order(data$id),] # order data by id and time

mean.formula <- ~fx1+fx2+x3
lv.formula <- ~1+x3
dat <- bmam::GenBinaryY(mean.formula = mean.formula, lv.formula = lv.formula,
                        beta=beta, Sigma=trueSigma, id=id, data=data, q=120,
                        Yname = "y")
names(dat)[c(8,9)] <- c("intercepts", "slopes")

knitr::kable(head(dat), digits = 4)
```

id	x1	fx1	x2	fx2	x3	y	intercepts	slopes
1	-0.330	-0.6311	-0.114	0.9849	0.567	1	2.4571	0.3752
1	0.818	-0.8243	0.500	0.0000	0.723	1	2.4571	0.3752
1	-0.177	-0.4378	0.597	0.8586	-0.360	1	2.4571	0.3752
1	-0.912	1.4146	-0.937	-0.5784	-0.111	1	2.4571	0.3752
1	0.527	0.4406	-0.046	0.4275	0.539	1	2.4571	0.3752

id	x1	fx1	x2	fx2	x3	y	intercepts	slopes
1	0.501	0.4980	0.648	1.2024	0.504	1	2.4571	0.3752

```r
x = seq(-1,1,length=1000)
par(mfrow = c(1,2))
plot(x, f1(x), type = "l", ylim = c(-2,2))
plot(x, f2(x), type = "l", ylim = c(-2,2))
```



Fit the BMAM

`BMAM` package is an extension of the `brmsmargins`(Wiley and Hedeker, 2022) which applies the post-hoc marginalization strategy to the case of linear associations(Hedeker et al., 2018).

We first load package.

```r
library(bmam)
```

Fit the underlying conditional model using `brms`

The BMAM is based on Bayesian GAMM, which is fitted by `brms::brm`.

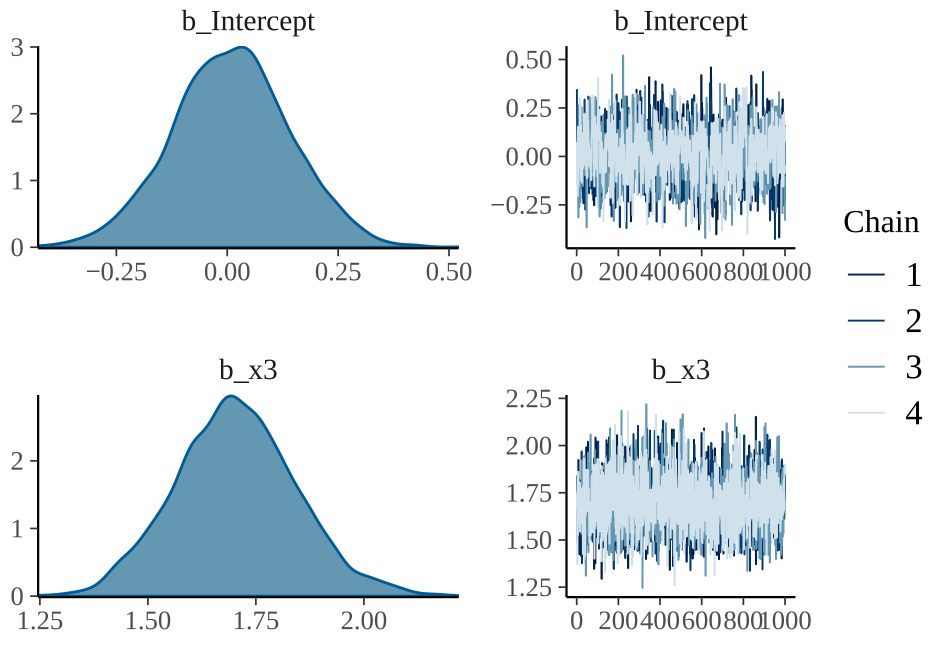
```r
library(brms)
model_brms <- brms::brm(brms::bf(y ~  x3 + s(x1) +
                                   s(x2) + (1+x3|id)),
                        data = dat, family = "bernoulli",
                        cores = 4, seed = 4321,
                        save_pars = save_pars(all = TRUE),
                        warmup = 1000, iter = 2000, chains = 4,
                        refresh=0, backend = "cmdstanr")
#> Running MCMC with 4 parallel chains...
#>
#> Chain 2 finished in 150.3 seconds.
#> Chain 4 finished in 150.4 seconds.
#> Chain 1 finished in 151.4 seconds.
#> Chain 3 finished in 154.0 seconds.
```

```
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 151.5 seconds.
#> Total execution time: 154.2 seconds.
```

The setting of `brm` function can be found in Bürkner (2017). We use `bf()` to specify a formula containing smooth terms and random effects. The GAMM is specified as a logistic model and we sample 4 chains with 2000 iterations (the first 1000 iterations as warm-up). It should be noted that our `BMAM` functions only support the `cmdstanr` backend, and we should specify `save_pars = save_pars(all = TRUE)` to let `brms` store all the parameters.

Then, we draw the trace plots of coefficients in linear terms to check the convergence and the density plots of the posterior samples.

```
plot(model_brms, par = c("b_Intercept", "b_x3"))
#> Warning: Argument 'pars' is deprecated. Please use 'variable' instead.
```



Fit the marginal model

To fit the marginal model, we use the `bmam` function.

```
bmam.fit <- bmam(object = model_brms)
```

Monte Carlo method is used to integrate out the random effects, and the default number of random draws is `k = 100`. The type of credible intervals (CI) can be specified using argument `CIType`. In this example, we report the Equal-Tailed Interval, ETI (default). Please see `bayestestR` package for more `CIType` options supported in this function. By default, the probability of CI is chosen as 95%.

Summarize model

Let's now summarize the fitted BMAM.

The summary function will return a list of data frames. Each data frame contains the summary of posterior distribution of the coefficients in linear terms, e.g. Intercept and X_3 in this example.

The first data frame is for the marginal estimates, and the second is for the conditional estimates (from the underlying Bayesian GAMM fitted by brms).

```
summary(bmam.fit)
#>
#> Family: bernoulli
#> Link function: logit
#>
#> y ~ x3 + s(x1) + s(x2) + (1 + x3 | id)
#>
#>  Marginal Model
#>    Parameter       M     Mdn     LL    UL   CI CIType
#> 1: Intercept -0.0259 -0.0272 -0.331 0.284 0.95    ETI
#> 2:        x3  1.0802  1.0790  0.864 1.300 0.95    ETI
#>
#>  Conditional Model
#>    Parameter       M     Mdn     LL   UL   CI CIType
#> 1: Intercept 0.00937 0.00981 -0.254 0.27 0.95    ETI
#> 2:        x3 1.70481 1.70185  1.435 2.00 0.95    ETI
```

The outputs are from the function `brmsmargins::bsummary`. For each parameter, the returned information include,

- M: the mean of the posterior samples
- Mdn: the median of the posterior samples
- LL: the lower limit of the credible interval
- UL: the upper limit of the credible interval
- CI: the probability of credible interval
- CIType: the type of credible interval

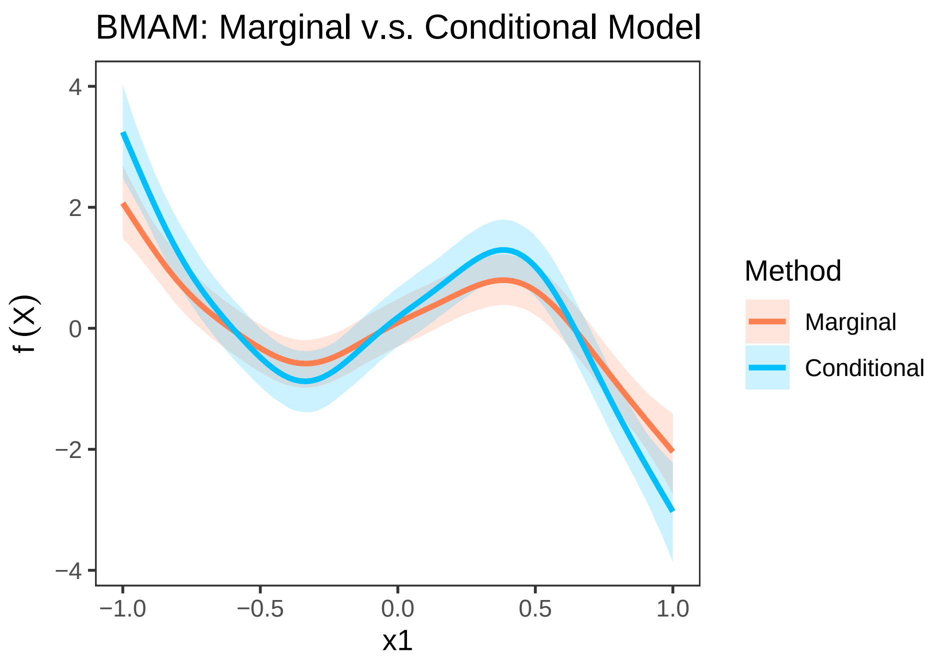
If interested in the smooth functions, we can set `plot.smooth` as `TRUE` to plot the estimated smooth functions, i.e. `summary(bmam.fit, plot.smooth = TRUE)`.

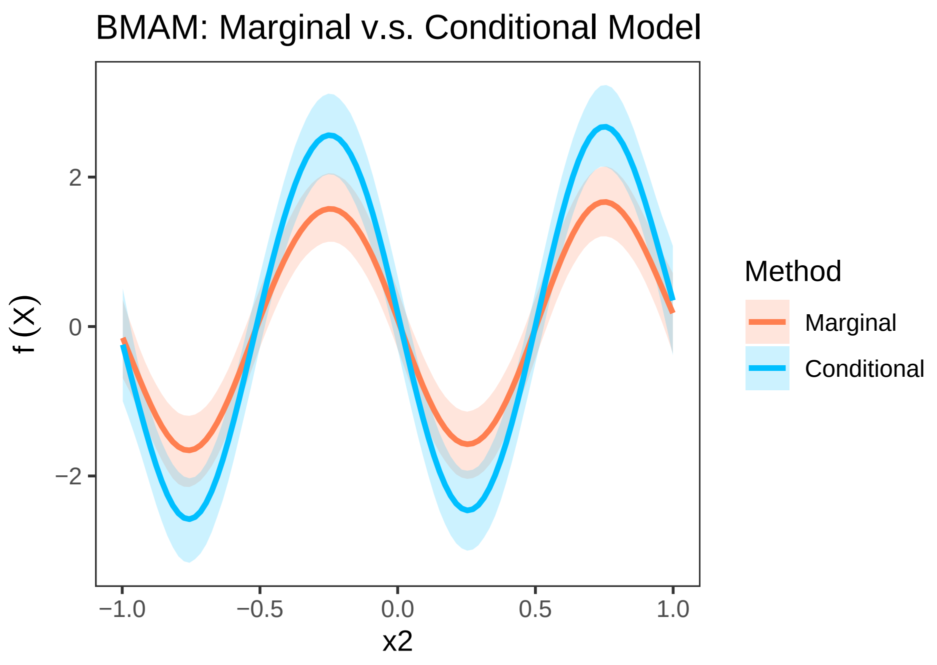
Visualize model

Beside the figures from `summary` function, we also provide a `plot` function to visualize the estimated smooth functions.

The `bmam` function will call the built-in function `generate_pred` in `bmam` package to generate the predicted data for illustrating the estimated smooth functions. The predicted data are generated according to the model and the data for model fitting. In this example, we use the simulated data to fit the model `bf(y ~ x3 + s(x1) + s(x2) + (1+x3|id))`. The function will generate a sequence with length 100 (default) from the minimum and maximum values of `x1` and a sequence with length is 100 (default) from the minimum and maximum values of `x1`.

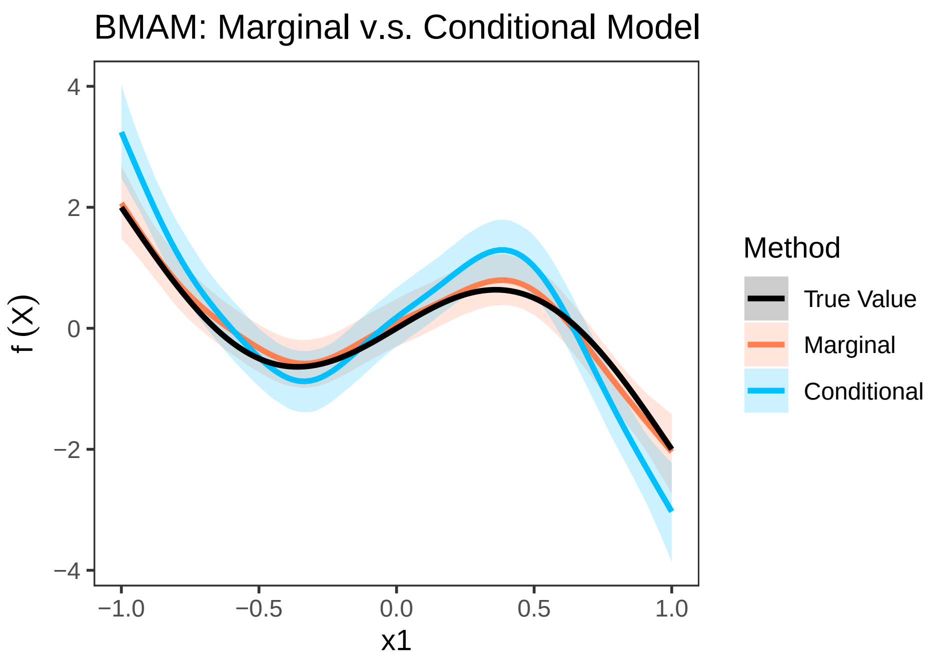
```
plot(bmam.fit)
```

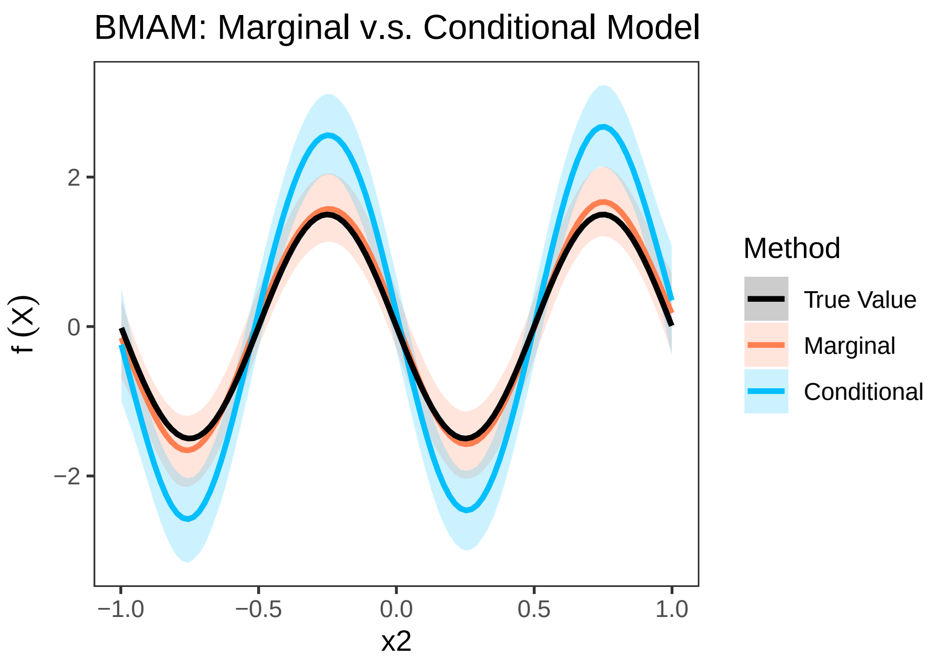




We could also provide the true forms of smooth functions to the `plot` function to compare the fitted values and true values,

```
plot(bmam.fit, smooth.function = c(f1,f2))
```





Compare with other models

The argument `compared.model` in `plot` function allows us to provide the other models compared with BMAM.

The supported models include,

- Marginal additive models by `mam` package
- Generalized additive models by `mgcv` package
- Bayesian generalized additive models by `brms` package

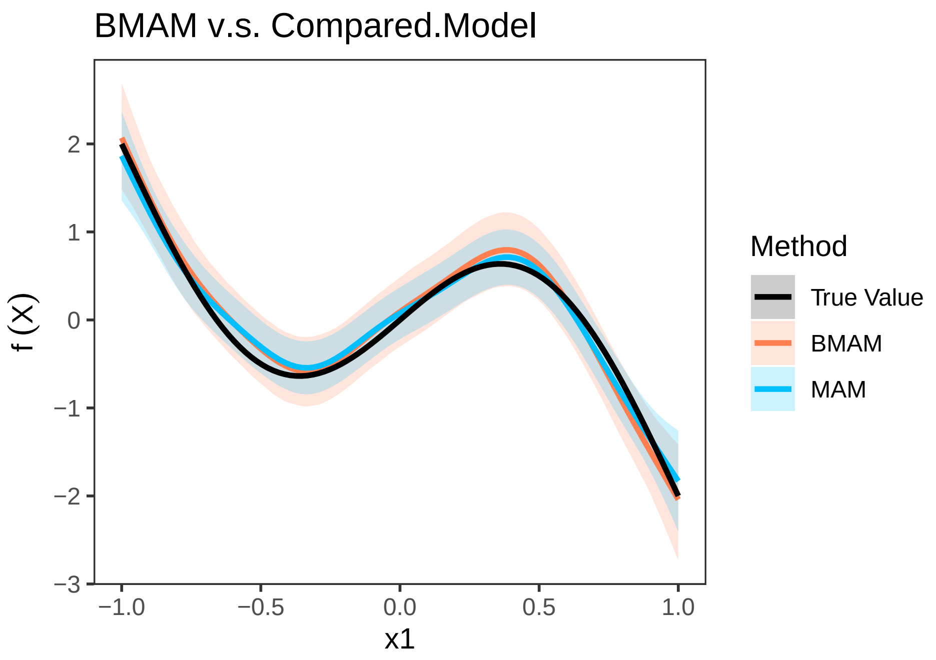
Marginal additive models

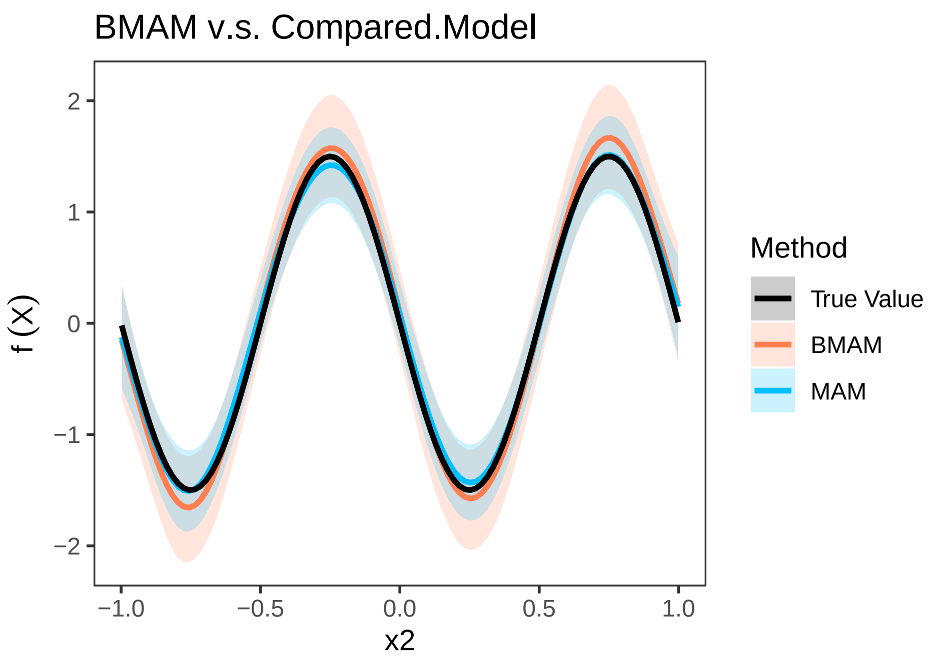
The frequentist marginal additive model can be fitted by `mam` function in `mam` package (McGee and Stringer, 2022). The predicted data in the `mam` should be the same as that we used in `bmam`. We can obtain the generated predicted data from the fitted `bmam` object by `bmam.fit$Preddat`.

```
library(mam)
library(mgcv)
```

```
themam <- mam(smooth = list(s(x1),s(x2)),
                   re = y ~ (1+x3|id),
                   fe = ~ x3,
                   dat = dat,
                   margdat = dat,
                   preddat = bmam.fit$Preddat,
                   control = mam_control(
                     method = 'trust',
                     varmethod = 1,
                     verbose = FALSE,
                     retcond = TRUE))
```

```
plot(bmam.fit, compared.model = themam, smooth.function = c(f1,f2))
```



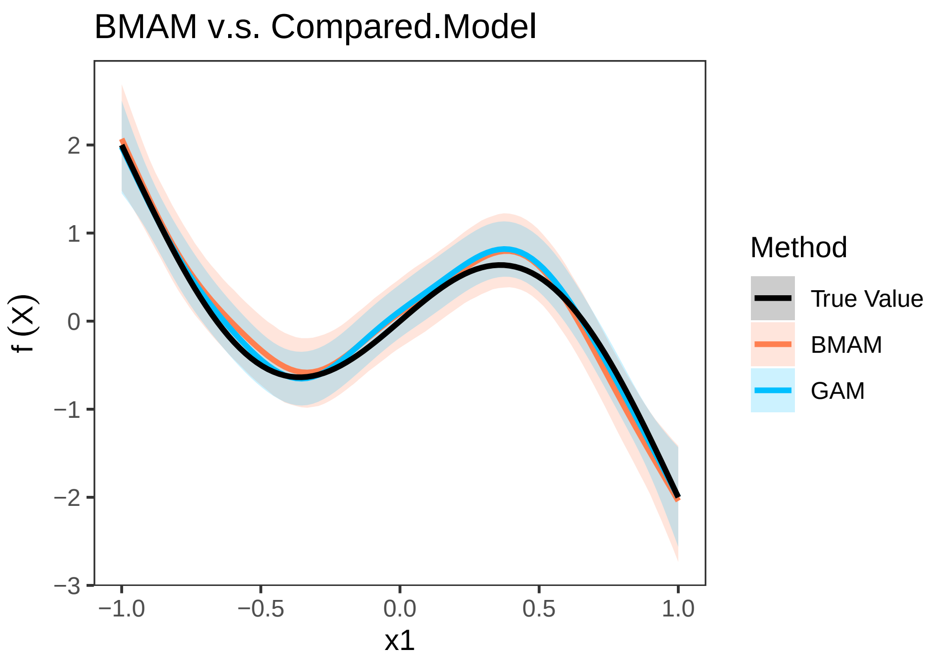


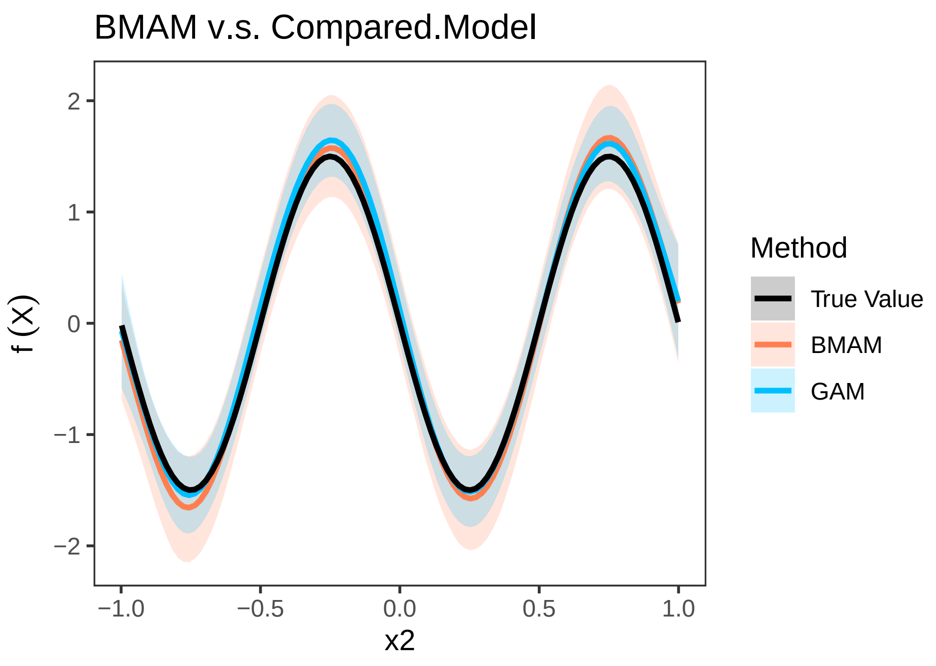
Generalized additive models

We could also compare the the models with GAM. The `plot` function now supports the generalized additive models by `mgcv` package and Bayesian generalized additive models by `brms` package.

```
gam <- gam(y ~  x3 + s(x1) + s(x2),
           data=dat,family=binomial(),method="REML")
```

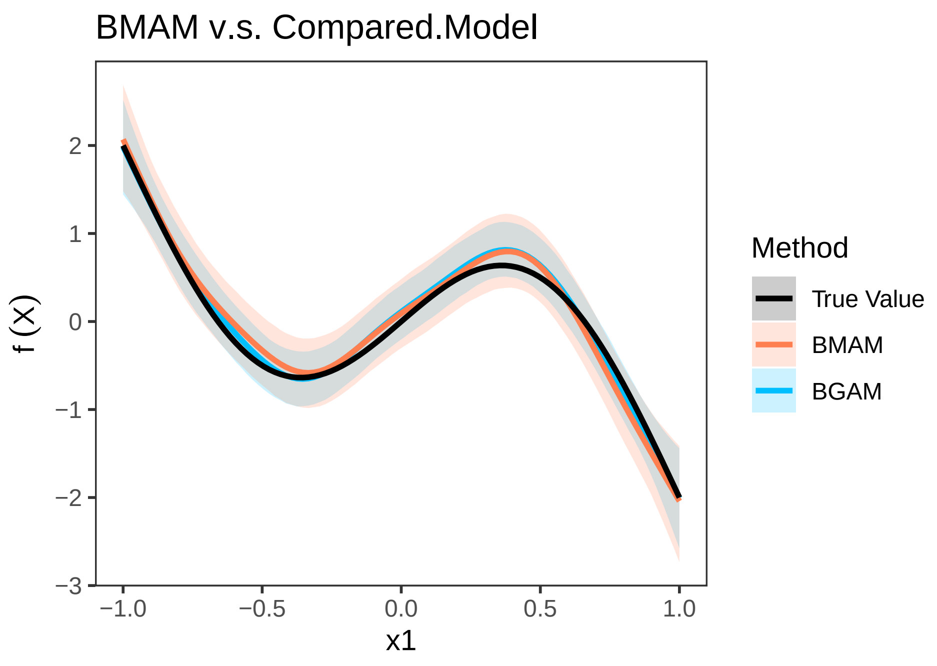
```
plot(bmam.fit, compared.model = gam, smooth.function = c(f1,f2))
```

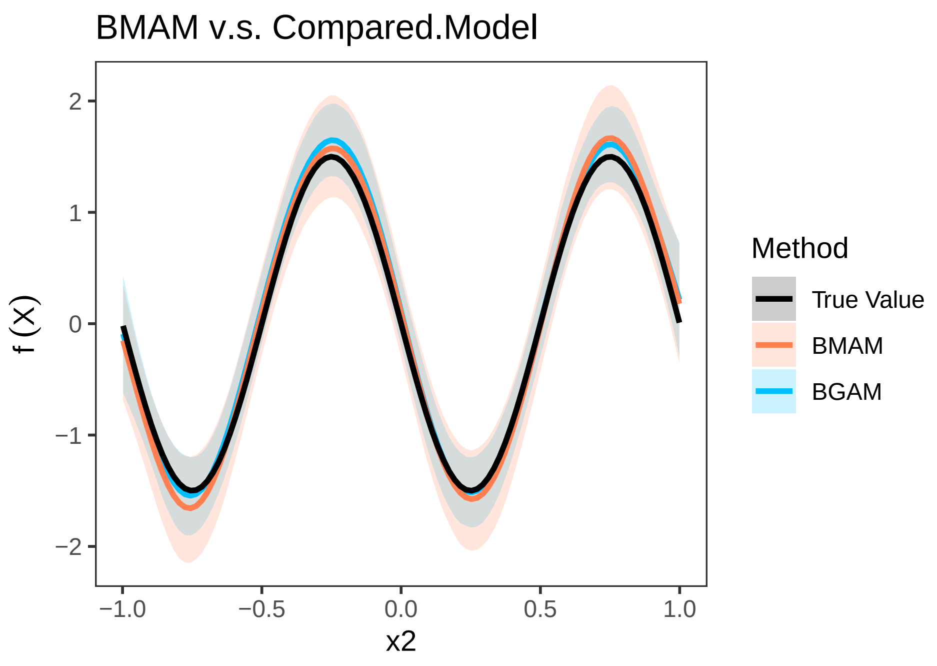




```
bgam <- brm(bf(y ~  x3 + s(x1) + s(x2)), data = dat,
            family = "bernoulli",
            cores = 4, seed = 4321, warmup = 1000,
            iter = 2000, chains = 4,
            refresh=0, backend = "cmdstanr")
#> Running MCMC with 4 parallel chains...
#>
#> Chain 2 finished in 79.1 seconds.
#> Chain 1 finished in 82.9 seconds.
#> Chain 4 finished in 85.2 seconds.
#> Chain 3 finished in 86.5 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 83.5 seconds.
#> Total execution time: 86.5 seconds.
```

```
plot(bmam.fit, compared.model = bgam, smooth.function = c(f1,f2))
```





References

Bürkner, P.-C. (2017). brms: An r package for bayesian multilevel models using stan. *Journal of statistical software*, 80:1–28.

Hedeker, D., du Toit, S. H., Demirtas, H., and Gibbons, R. D. (2018). A note on marginalization of regression parameters from mixed models of binary outcomes. *Biometrics*, 74(1):354–361.

McGee, G. and Stringer, A. (2022). Flexible marginal models for dependent data. *arXiv preprint arXiv:2204.07188*.

Wiley, J. F. and Hedeker, D. (2022). *brmsmargins: Bayesian Marginal Effects for 'brms' Models*. https://joshuawiley.com/brmsmargins/, https://github.com/JWiley/brmsmargins.