Introduction to Bayesian Marginal Additive Models

Tianyi Pan

2022-05-16

Contents

In this document, we illustrate R functions for Bayesian marginal additive models (BMAM).

Brief overview of Bayesian marginal additive models

Heagerty and Zeger (2000) presented the marginal models in which the marginal mean, rather than the conditional mean given random effects, is regressed on covariates. Marginal models are useful when we are interested in associations averaged across a population of clusters. If the covariate-outcome association is unknown, we could use additive models to measure the potentially non-linear relationships. McGee and Stringer (2022) proposed a marginal additive model (MAM) for modeling dependent data with non-linear population-averaged associations.

We propose a Bayesian marginal additive model (BMAM) extending the marginal additive model (MAM) into a Bayesian context.

Bayesian conditional models

We consider a Generalized Additive Mixed Model (GAMM),

$$g\left\{\mu^{\mathrm{C}}\left(\boldsymbol{x}_{ij}, \boldsymbol{z}_{ij} \mid \boldsymbol{u}_i\right)\right\} = \sum_{l=1}^{p} f_l^{\mathrm{C}}\left(x_{ijl}\right) + \boldsymbol{z}_{ij}^{\mathrm{T}} \boldsymbol{u}_i$$

where $i = 1, \cdots, N$ indexes clusters and $j = 1, \cdots, n_i$ indexes units within clusters. Random effects $\boldsymbol{u}_i$ follow a normal distribution with mean 0 and covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{\Lambda}(\boldsymbol{\sigma}) \boldsymbol{\Omega} \boldsymbol{\Lambda}(\boldsymbol{\sigma})$, where $\boldsymbol{\Lambda}(\boldsymbol{\sigma})$ denotes the diagonal matrix with diagonal elements $\boldsymbol{\sigma}$.

$f_l^{\mathrm{C}}(x) = \sum_{q=1}^{d_l} b_{lq}^{\mathrm{C}}(x)\alpha_{lq}^{\mathrm{C}}$ are unknown smooth functions. Each $\boldsymbol{\alpha}_l^{\mathrm{C}}$ is associated with a penalty $\mathcal{P}_{\tau_l^{\mathrm{C}}}\left(\boldsymbol{\alpha}_l^{\mathrm{C}}; \boldsymbol{S}_l\right) \propto \exp\left(-\tau_l \boldsymbol{\alpha}_l^{\mathrm{C}^{\mathrm{T}}} \boldsymbol{S}_l \boldsymbol{\alpha}_l^{\mathrm{C}}\right)$.

According to Wood (2006), $\sum_{q=1}^{d_l} b_{lq}^{\mathrm{C}}(x)\alpha_{lq}^{\mathrm{C}}$ can be written as $\sum_{q=1}^{d_l} b_{lq}^{*\mathrm{C}}(x)\alpha_{lq}^{*\mathrm{C}}$. Then the penalty can be termed as a prior distribution,

$$\alpha_{lk}^{*\mathrm{C}} \sim \mathrm{N}(0, \frac{1}{\tau_l}), k = 3, \cdots, d_l,$$

where $\left[b_{l1}^{*\mathrm{C}}(x), b_{l2}^{*\mathrm{C}}(x), \cdots, b_{ld_l}^{*\mathrm{C}}(x)\right] = \left[b_{l1}^{\mathrm{C}}(x), b_{l2}^{\mathrm{C}}(x), \cdots, b_{ld_l}^{\mathrm{C}}(x)\right] \mathbf{D}_{l+}^{-1}$, $\mathbf{D}_{l+} = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0} \\ & \mathbf{D}_l \end{bmatrix}$. and $\mathbf{D}_l$ is a $(d_l - 2) \times d_l$ matrix with $\mathbf{D}_l^{\mathrm{T}}\mathbf{D}_l = S_l$

Thus, the GAMM can be rewritten as,

$$\begin{aligned} g\left\{\mu^{\mathrm{C}}\left(\boldsymbol{x}_{ij}, \boldsymbol{z}_{ij} \mid \boldsymbol{u}_i\right)\right\} &= \sum_{l=1}^{p}\sum_{q=1}^{d_l} b_{lq}^{*\mathrm{C}}(x)\alpha_{lq}^{*\mathrm{C}} + \boldsymbol{z}_{ij}^{\mathrm{T}}\boldsymbol{u}_i, \\ \boldsymbol{u}_i &\sim \mathrm{N}(0, \boldsymbol{\Lambda}\left(\boldsymbol{\sigma}\right)\boldsymbol{\Omega}\boldsymbol{\Lambda}\left(\boldsymbol{\sigma}\right)), \\ \alpha_{lk}^{*\mathrm{C}} &\sim \mathrm{N}(0, \frac{1}{\tau_l}), k = 3, \cdots, d_l. \end{aligned}$$

Besides the two parameters $\boldsymbol{u}_i$ and $\alpha_{lk}^{*\mathrm{C}}, k = 3, \cdots, d_l$, in a Bayesian model, we need to assign prior distributions for $\alpha_{l1}^{*\mathrm{C}}, \alpha_{l2}^{*\mathrm{C}}$, τ_l, $\boldsymbol{\sigma}$ and $\boldsymbol{\Omega}$.

Our `bmam` functions are built on `brms` package. The prior distributions can be set in `brms` function. By default, $\alpha_{l1}^{*\mathrm{C}}, \alpha_{l2}^{*\mathrm{C}}$ have an improper flat prior, and $\sqrt{1/\tau_l}$, the standard deviation in prior distribution of $\boldsymbol{u}_i$, has a half student-t prior with 3 degree of freedom. The prior distribution of $\boldsymbol{\sigma}$ is also half student-t prior with 3 degree of freedom, and $\Omega \sim \mathrm{LKJ}(1)$.

Bayesian marginal additive models

The Bayesian marginal additive model (BMAM) is,

$$g\left\{\mu^{\mathrm{M}}\left(\boldsymbol{x}_{ij}\right)\right\} = f^{\mathrm{M}}\left(\boldsymbol{x}_{ij}\right)$$

where $f^{\mathrm{M}}\left(\boldsymbol{x}_{ij}\right) = \sum_{l=1}^{p} f_l^{\mathrm{M}}\left(x_{ijl}\right) = \sum_{l=1}^{p}\sum_{q=1}^{Q_l} b_{lq}^{\mathrm{M}}(x)\alpha_{lq}^{\mathrm{M}}$. The inference on parameters α_{lq}^{M} is based on the Bayesian conditional model through post-hoc marginalization (Hedeker et al. 2018). More details about the relationship between marginal additive models and conditional models can be found in McGee and Stringer (2022).

In practice, we could draw the posterior samples in conditional models. For each posterior sample, an estimate of α_{lq}^{M} could be calculated. Assuming H samples are drawn from posterior distribution in a conditional model, we could obtain H estimates of α_{lq}^{M}, and then make an inference on α_{lq}^{M}.

Basic example of Bayesian marginal additive models

We use the function `SimData` to generate some data, to illustrate the main features of our R functions for BMAM. The data are generated according to the set-up in McGee and Stringer (2022). 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.

We generate three covariates $\boldsymbol{x}_{ij} = (x_{ij1}, x_{ij2}, x_{ij3})^{\mathrm{T}}$ where $x_{ijp} \sim \mathrm{Unif}(-1, 1)$ for $p = 1, 2, 3$. After assuming the forms of smooth functions $f_1^{\mathrm{M}}(\cdot)$ and $f_2^{\mathrm{M}}(\cdot)$, we could generate Y_{ij} according to the marginal model,

$$\mathrm{logit}\left\{\mu^{\mathrm{M}}\left(\boldsymbol{x}_{ij}\right)\right\} = f_1^{\mathrm{M}}\left(x_{ij1}\right) + f_2^{\mathrm{M}}\left(x_{ij2}\right) + \beta_3 \cdot x_{ij3}.$$

In addition, we have the following random intercepts and slopes dependence structure,

$$\begin{aligned} \text{logit}\left\{\mu^{\text{C}}\left(\boldsymbol{x}_{ij} \mid u_i\right)\right\} &= \Delta\left(\boldsymbol{x}_{ij}\right) + u_{i0} + u_{i1} \cdot x_{ij3} \\ \begin{bmatrix} u_{i0} \\ u_{i1} \end{bmatrix} &\sim MVN\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 4 & 1 \\ 1 & 1 \end{bmatrix}\right) \end{aligned}$$

Here we set $\beta_3 = 0$, the number of cluster $N = 100$ and the number of units within clusters $n_i = 10, i = 1, \ldots, N$.

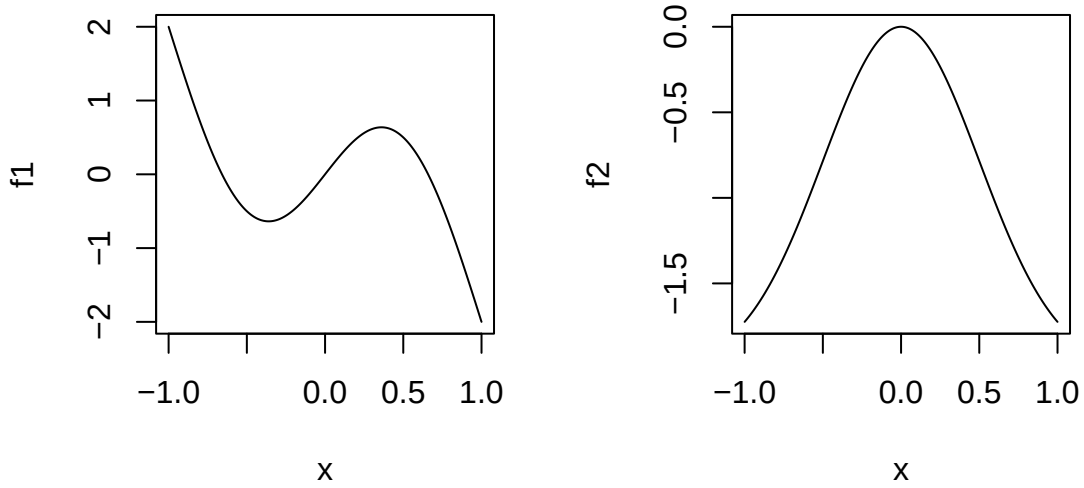
```
set.seed(4321)
simdata <- SimData(K = 100, Nk = 10)
dat <- simdata$data
fun <- simdata$f

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

id	intercepts	slopes	x1	x2	x3	fx1	fx2	y
1	0.0042	0.7361	-0.330	0.961	0.170	-0.6311	-1.6801	0
1	0.0042	0.7361	0.818	-0.099	0.043	-0.8243	-0.0387	0
1	0.0042	0.7361	-0.177	-0.329	0.236	-0.4378	-0.3883	1
1	0.0042	0.7361	-0.912	-0.377	-0.130	1.4146	-0.4935	1
1	0.0042	0.7361	0.527	0.715	0.582	0.4406	-1.2772	0
1	0.0042	0.7361	0.501	0.095	0.521	0.4980	-0.0357	0

where `simdata$data` is the generated dataset and `simdata$f` is the smooth functions $f_1^{\text{M}}(\cdot)$ and $f_2^{\text{M}}(\cdot)$.

```
x = seq(-1,1,length=1000)
f1 <- fun[[1]](x)
f2 <- fun[[2]](x)
par(mfrow = c(1,2))
plot(x, f1, type = "l")
plot(x, f2, type = "l")
```



Fit BMAM

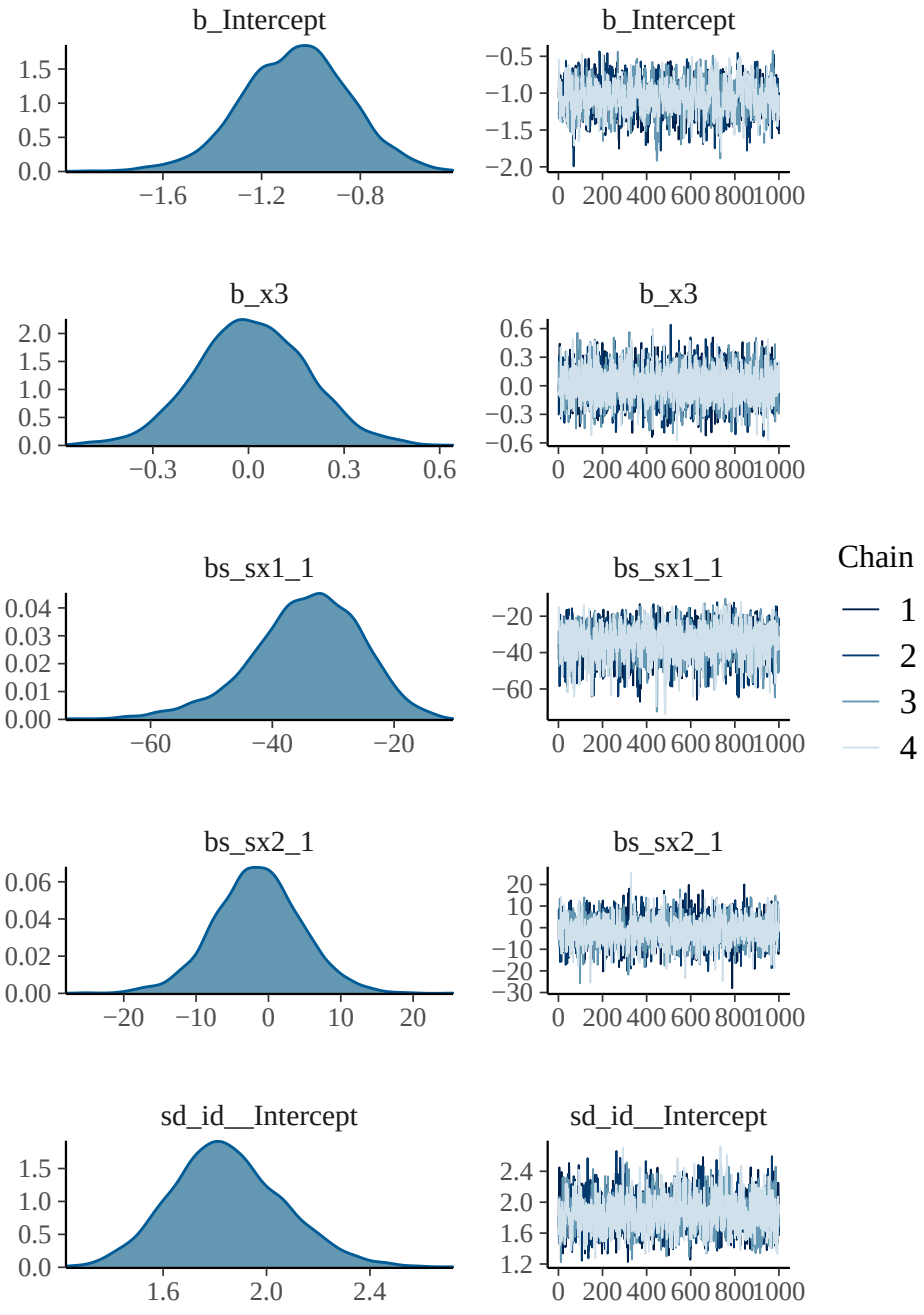
The BMAM is based on Bayesian GAMM, which could fitted by `brm` function in `brms` package.

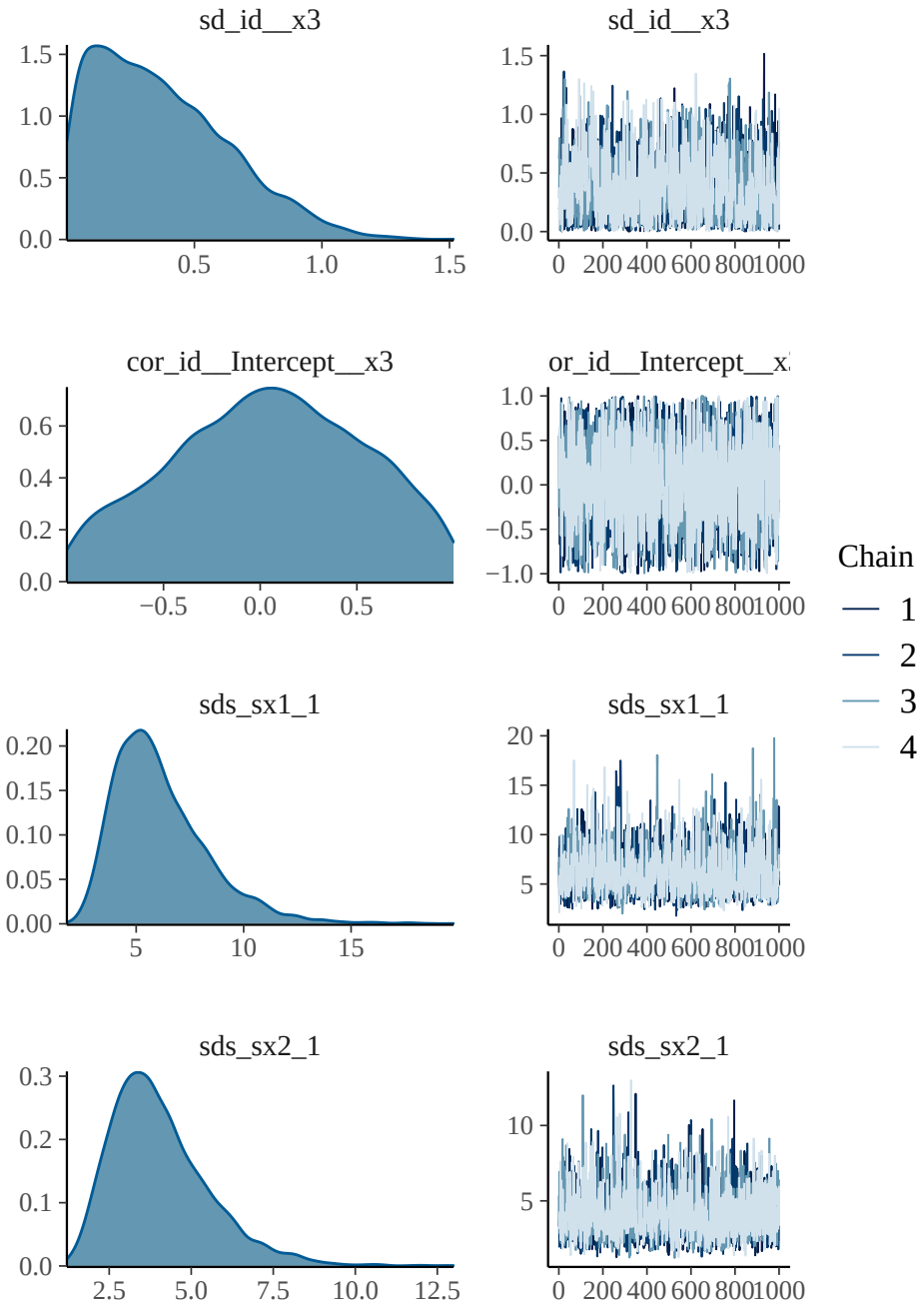
```
library(brms)
model_brms <- brm(bf(y ~  x3 + s(x1) + s(x2) + (1+x3|id)),
                  data = dat, family = "bernoulli",
                  cores = 4, seed = 4321,
                  warmup = 1000, iter = 2000, chains = 4,
                  refresh=0, backend = "cmdstanr")
#> Running MCMC with 4 parallel chains...
#>
#> Chain 3 finished in 34.4 seconds.
#> Chain 4 finished in 34.8 seconds.
#> Chain 1 finished in 35.3 seconds.
#> Chain 2 finished in 36.2 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 35.2 seconds.
#> Total execution time: 36.5 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 are warm-up). It should be noted that our `BMAM` functions only support the `cmdstanr` backend.

Then, we draw the trace plots to check the convergence in Hamiltonian Monte Carlo in `brms` and the density plots of the posterior samples.

```
plot(model_brms)
```


b_Intercept
b_Intercept
b_x3
b_x3
bs_sx1_1
bs_sx1_1
bs_sx2_1
bs_sx2_1
sd_id__Intercept
sd_id__Intercept
Chain
1
2
3
4



`BMAM` functions are built on `brmsmargins` package. We firstly load the dependent packages,

```
library(brmsmargins)
library(data.table)
library(dplyr)
library(ggplot2)
```

To fit the BMAM, we use the `bmam` function.

```
bmam.fit <- bmam(object = model_brms, k=100, CIType="ETI", CI = 0.95)
```

Monte Carlo method is used to integrate out the random effects, and we set the random draws as `k = 100` (default). The type of credible intervals (CI) can be specified by `CIType`. In this example, we use Equal-Tailed Interval, ETI (default). Please see here for more `CIType` options supported in this function. 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 dataframe is for the estimates in marginal model (BMAM), and the second is for the estimates in conditional model (Bayesian GAMM).

```
bmam.summary <- 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 -0.6994 -0.6965 -1.082 -0.332 0.95    ETI
#> 2:        x3  0.0115  0.0109 -0.218  0.242 0.95    ETI
#>
#>  Conditional Model
#>    Parameter       M      Mdn     LL     UL   CI CIType
#> 1: Intercept -1.0624 -1.05754 -1.503 -0.646 0.95    ETI
#> 2:        x3  0.0111  0.00838 -0.332  0.358 0.95    ETI
```

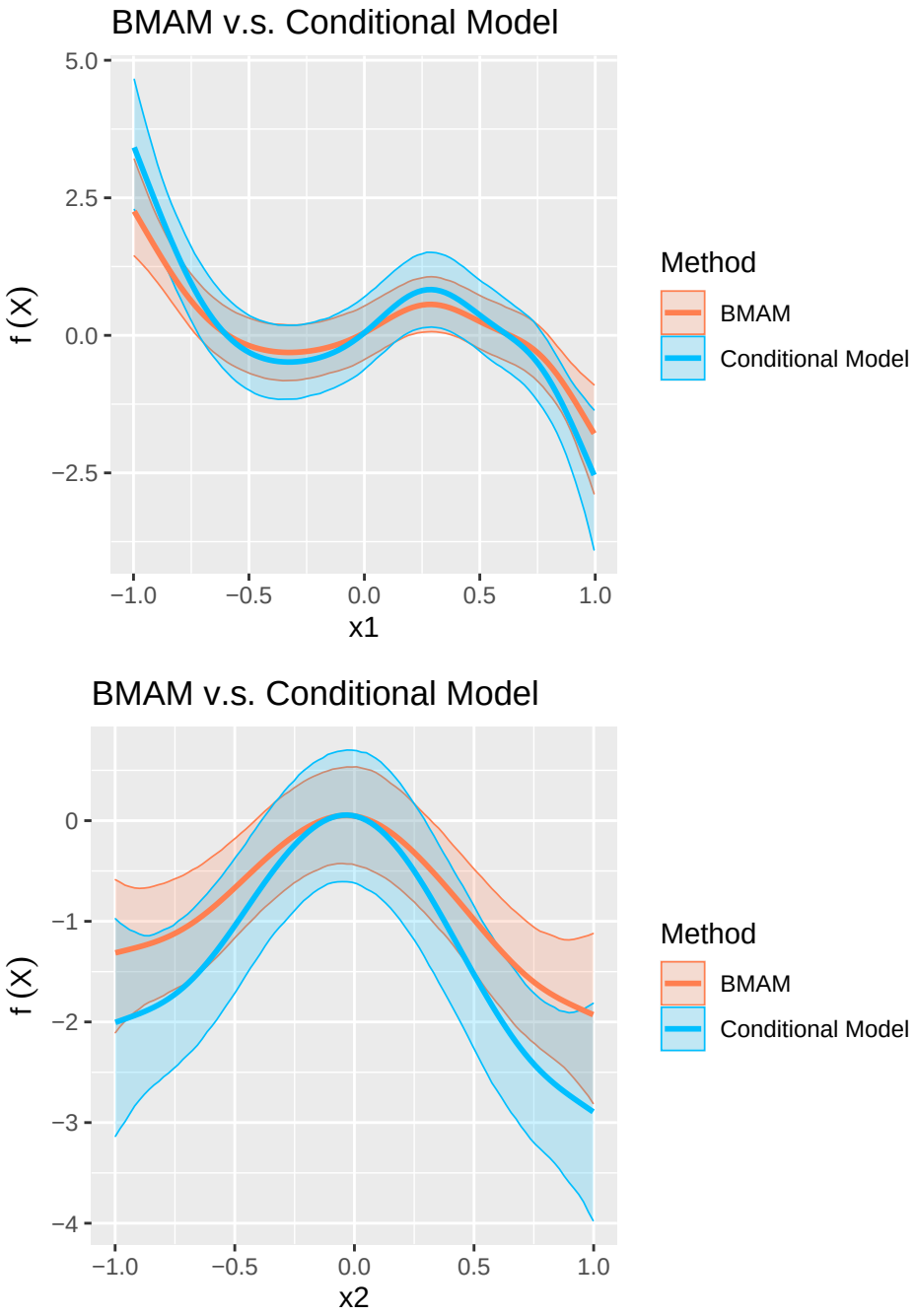
For each parameter, the returned information include,

- 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 we are also interested in the smooth functions, we can set `plot_smooth` as `TRUE` to plot the estimated smooth functions,

```
bmam.summary <- summary(bmam.fit, plot_smooth = TRUE)
#>
#> 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 -0.6994 -0.6965 -1.082 -0.332 0.95    ETI
#> 2:        x3  0.0115  0.0109 -0.218  0.242 0.95    ETI
#>
#>  Conditional Model
#>    Parameter       M      Mdn     LL     UL   CI CIType
#> 1: Intercept -1.0624 -1.05754 -1.503 -0.646 0.95    ETI
#> 2:        x3  0.0111  0.00838 -0.332  0.358 0.95    ETI
```



In the figures, we demontrate the point estimates (mean of the posterior samples) and 95% CI (ETI). The red one denotes the estimates from BMAM. The blue one denotes the estimates from Bayesian GAMM (conditional model).

The results from `summary` function also provide the summary of coefficients in smooth terms α_l^{M} in BMAM and smooth terms α_l^{C} in Bayesian GAM.

The default dimension of coefficients α_l^{M} (or α_l^{C}) is 10, i.e. $Q_1 = Q_2 = 10$ in this example. The first parameter α_{l1}^{M} (or α_{l1}^{C}) is the intercept, and we only report the coefficients $\alpha_{lq}^{\mathrm{M}}, \alpha_{lq}^{\mathrm{C}}, q = 2, 3, \cdots, Q_l$.

The coefficients in smooth terms in BMAM,

```
knitr::kable(bmam.summary$BMAM$Smooth[[1]], digits = 4)
```

Parameter	M	Mdn	LL	UL	CI	CIType
Xs_1_alpha	-23.929	-23.3924	-38.90	-12.81	0.95	ETI
Zs_1_1_alpha_1	-1.599	-1.2577	-10.65	5.50	0.95	ETI
Zs_1_1_alpha_2	-0.719	-0.7923	-4.89	3.65	0.95	ETI
Zs_1_1_alpha_3	8.046	7.9326	5.16	11.59	0.95	ETI
Zs_1_1_alpha_4	-0.102	-0.0951	-2.35	2.12	0.95	ETI
Zs_1_1_alpha_5	-0.442	-0.5163	-5.36	4.73	0.95	ETI
Zs_1_1_alpha_6	-0.706	-0.6397	-5.86	3.73	0.95	ETI
Zs_1_1_alpha_7	-2.127	-1.8284	-9.69	3.95	0.95	ETI
Zs_1_1_alpha_8	-4.862	-4.3483	-14.45	1.82	0.95	ETI

```
knitr::kable(bmam.summary$BMAM$Smooth[[2]], digits = 4)
```

Parameter	M	Mdn	LL	UL	CI	CIType
Xs_2_alpha	-0.9211	-0.9319	-9.41	7.45	0.95	ETI
Zs_2_1_alpha_1	0.2962	0.2211	-4.93	5.69	0.95	ETI
Zs_2_1_alpha_2	-1.0763	-0.9099	-5.33	2.29	0.95	ETI
Zs_2_1_alpha_3	0.1172	0.1285	-2.17	2.38	0.95	ETI
Zs_2_1_alpha_4	-4.7807	-4.6938	-7.56	-2.48	0.95	ETI
Zs_2_1_alpha_5	-0.0794	-0.0918	-4.14	4.12	0.95	ETI
Zs_2_1_alpha_6	0.1061	0.0818	-3.96	4.24	0.95	ETI
Zs_2_1_alpha_7	0.0845	0.0804	-5.07	5.15	0.95	ETI
Zs_2_1_alpha_8	0.2724	0.2286	-4.77	5.53	0.95	ETI

The coefficients in smooth terms in Bayesian GAMM (conditional model),

```
knitr::kable(bmam.summary$Conditional_Model$Smooth[[1]], digits = 4)
```

Parameter	M	Mdn	LL	UL	CI	CIType
Xs_1_alpha	-34.419	-33.724	-54.71	-18.65	0.95	ETI
Zs_1_1_alpha_1	-1.586	-1.203	-14.91	9.10	0.95	ETI
Zs_1_1_alpha_2	-1.614	-1.703	-7.57	4.44	0.95	ETI
Zs_1_1_alpha_3	11.741	11.664	7.68	16.62	0.95	ETI
Zs_1_1_alpha_4	-0.403	-0.388	-3.62	2.76	0.95	ETI
Zs_1_1_alpha_5	-0.592	-0.696	-7.83	7.18	0.95	ETI
Zs_1_1_alpha_6	-0.499	-0.407	-7.83	6.04	0.95	ETI
Zs_1_1_alpha_7	-2.553	-2.141	-13.67	6.69	0.95	ETI
Zs_1_1_alpha_8	-7.081	-6.349	-21.53	2.75	0.95	ETI

```
knitr::kable(bmam.summary$Conditional_Model$Smooth[[2]], digits = 4)
```

Parameter	M	Mdn	LL	UL	CI	CIType
Xs_2_alpha	-1.4728	-1.4919	-13.55	10.71	0.95	ETI
Zs_2_1_alpha_1	0.3013	0.2706	-7.82	8.41	0.95	ETI
Zs_2_1_alpha_2	-1.6142	-1.3643	-7.79	3.19	0.95	ETI
Zs_2_1_alpha_3	0.1438	0.1565	-3.10	3.50	0.95	ETI
Zs_2_1_alpha_4	-7.3035	-7.2037	-11.24	-3.96	0.95	ETI
Zs_2_1_alpha_5	-0.2824	-0.2975	-6.35	5.84	0.95	ETI
Zs_2_1_alpha_6	-0.0276	-0.0578	-6.03	6.16	0.95	ETI
Zs_2_1_alpha_7	0.3910	0.3588	-7.17	7.97	0.95	ETI
Zs_2_1_alpha_8	-0.1244	-0.1869	-7.86	7.40	0.95	ETI

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` to generate the predicted data to illustrate 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 `simdata$data` to fit the model `bf(y ~ x3 + s(x1) + s(x2) + (1+x3|id))`. The function will generate a sequence with length 100 (default) from the minimum value of `x1` to the maximum value of `x1` in `simdata$data` and a sequence with length is 100 (default) from the minimum value of `x2` to the maximum value of `x2` in `simdata$data`. The length of sequence can be set as the other values by argument `length`, for example,

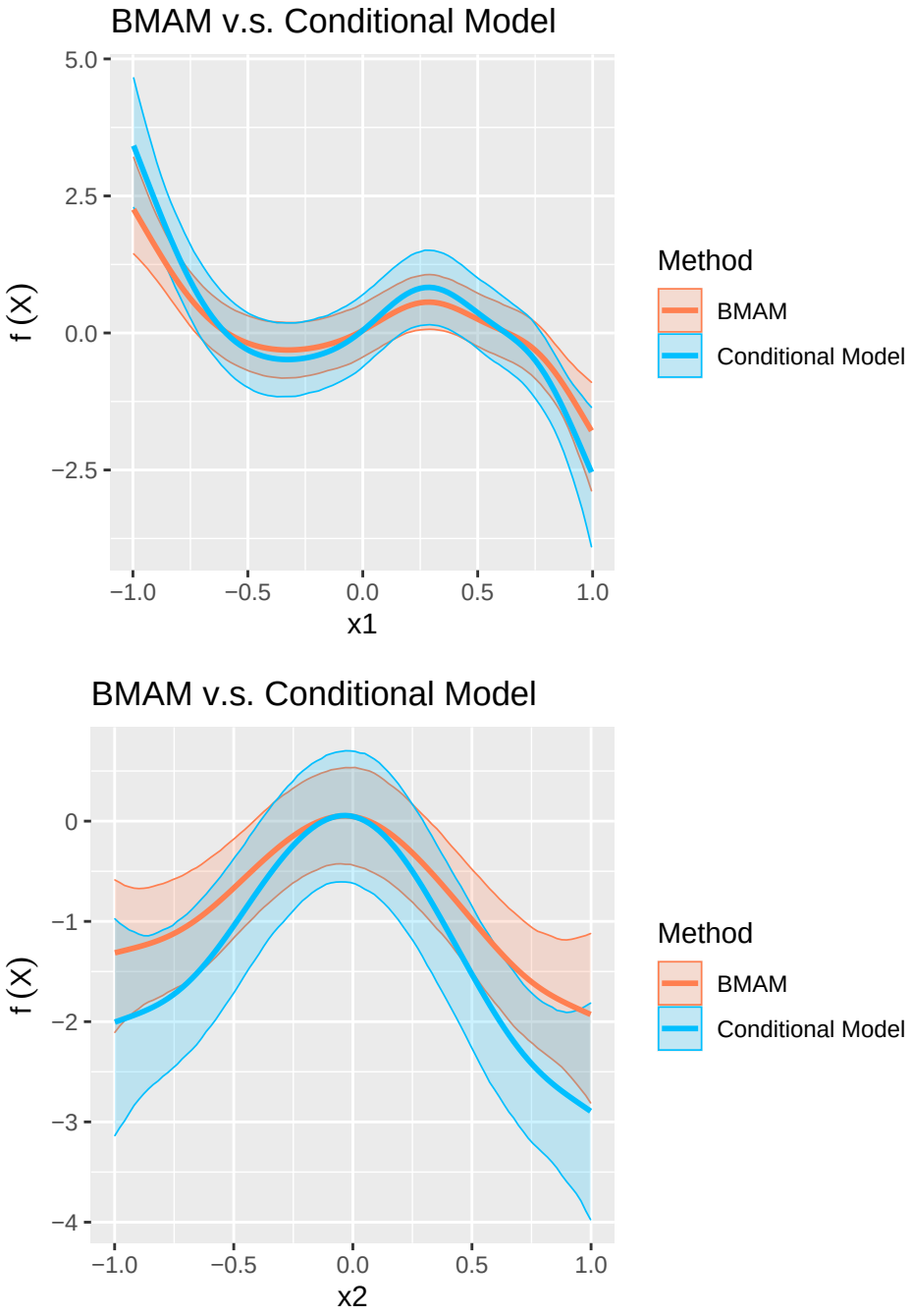
```
bmam(object = model_brms, k=100, CIType="ETI", CI = 0.95,
     length = 200)
```

In addition, users can also provide a predicted data, for example,

```
bmam(object = model_brms, k=100, CIType="ETI", CI = 0.95,
     preddat = user.preddat)
```

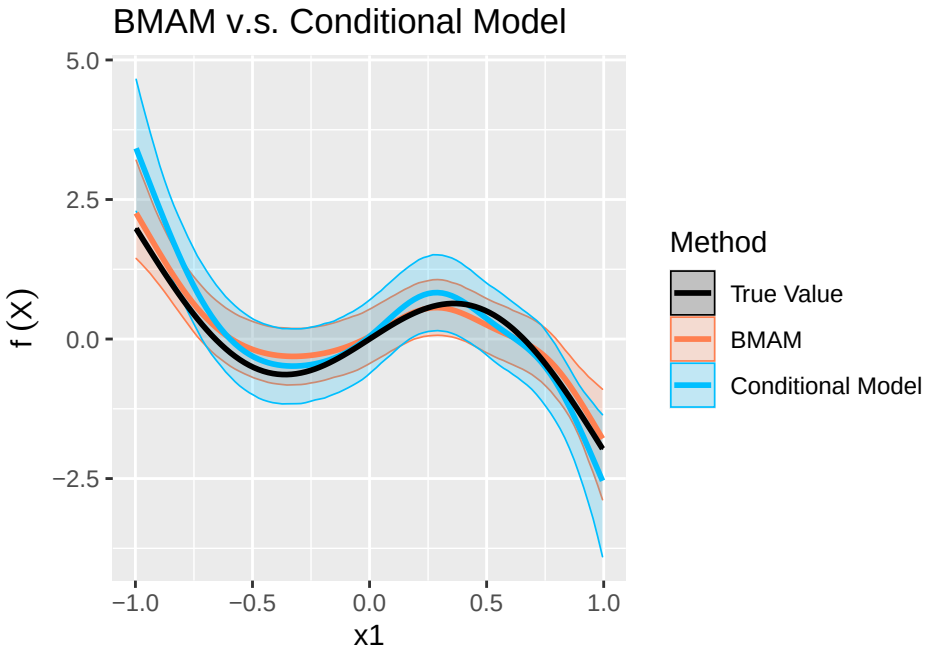
The `plot` function will show the plots of the estimated smooth functions in BMAM and the conditional model (Bayesian GAMM). If users want to plot the BMAM only, please add an argument `conditional = FALSE`.

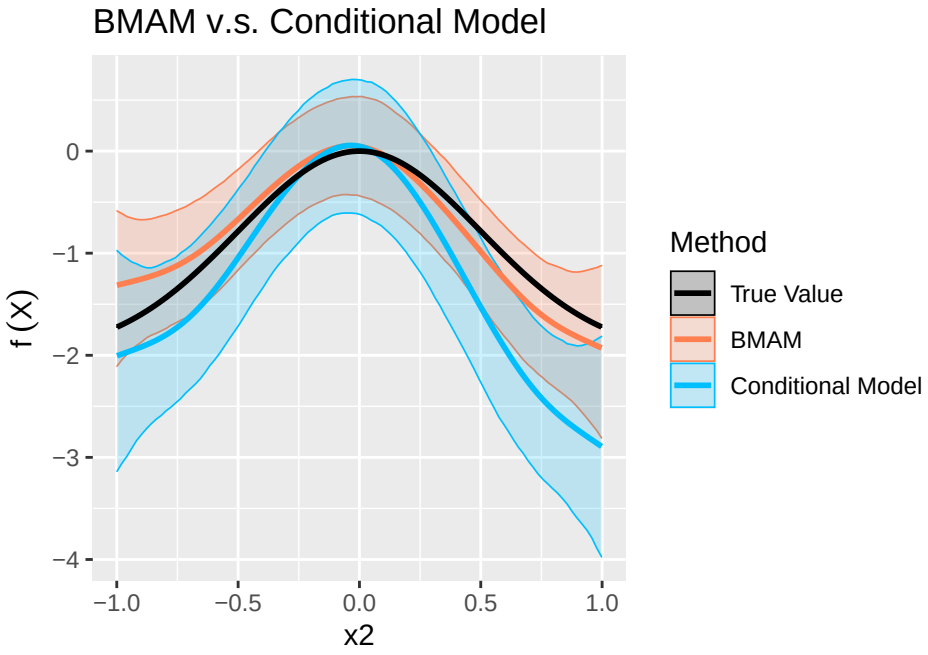
```
plot(bmam.fit)
```



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

```
plot(bmam.fit, smooth.function = simdata$f)
```





Compare with other models

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

The supported models include

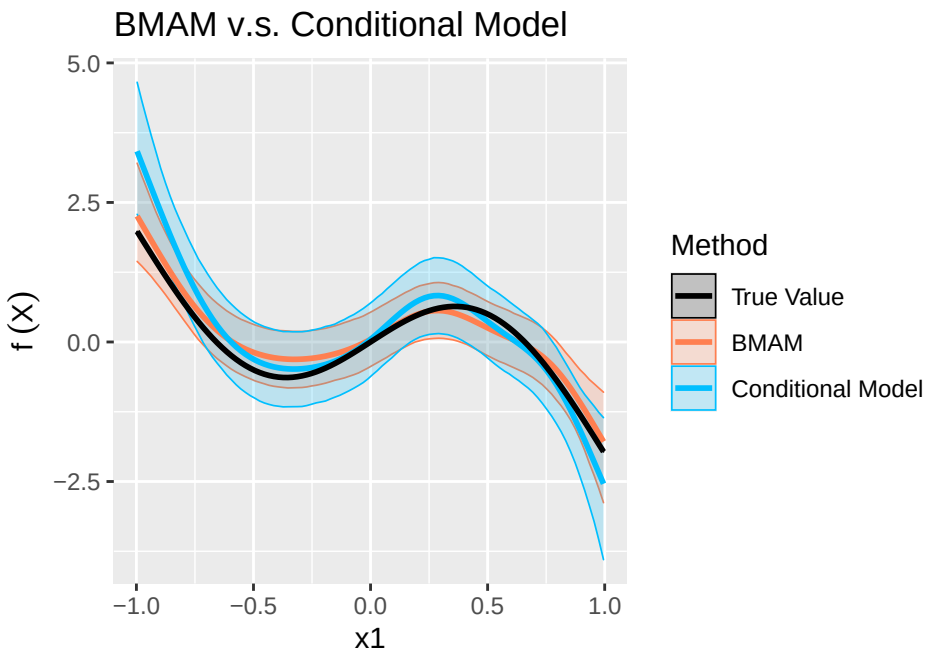
- Marginal additive models by `mam` package
- Generalized additive models by `mgcv` package
- Bayesian generalized additive models by `brms` package

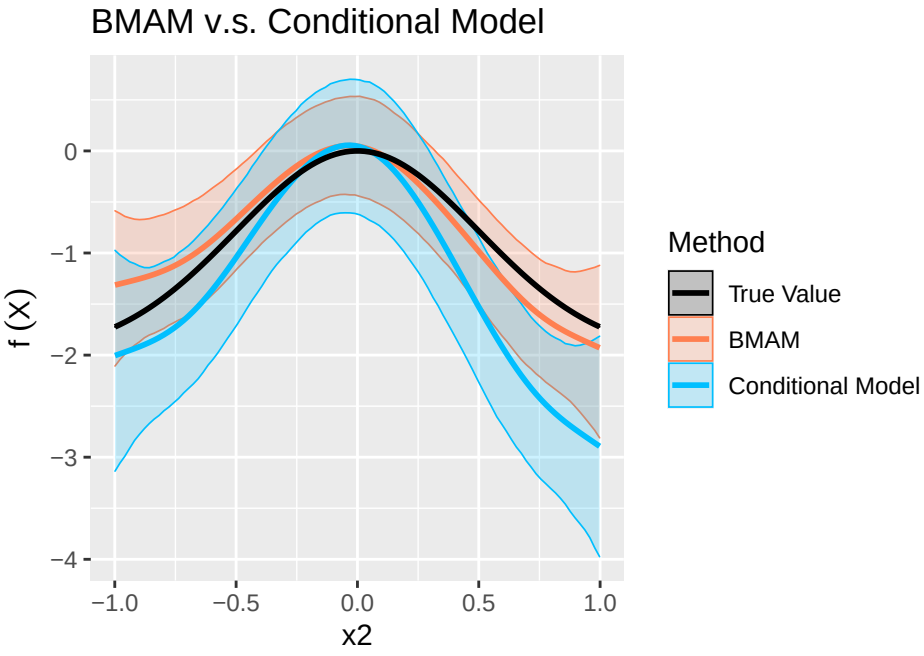
Marginal additive models The frequentist marginal additive model can be fitted by `mam` function in `mam` package (McGee and Stringer 2022). The predicted data in the `mam` should be the same as that we used in `bmam`. We can obtain the generated predicted data from the fitted `bmam` object by `bmam.fit$Preddat`.

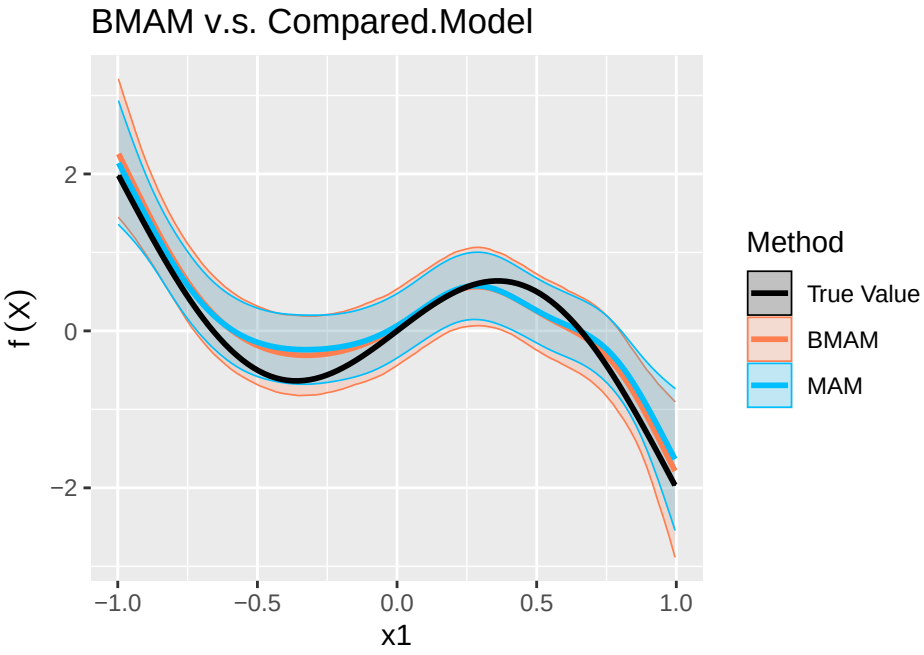
```
library(mam)
library(mgcv)
```

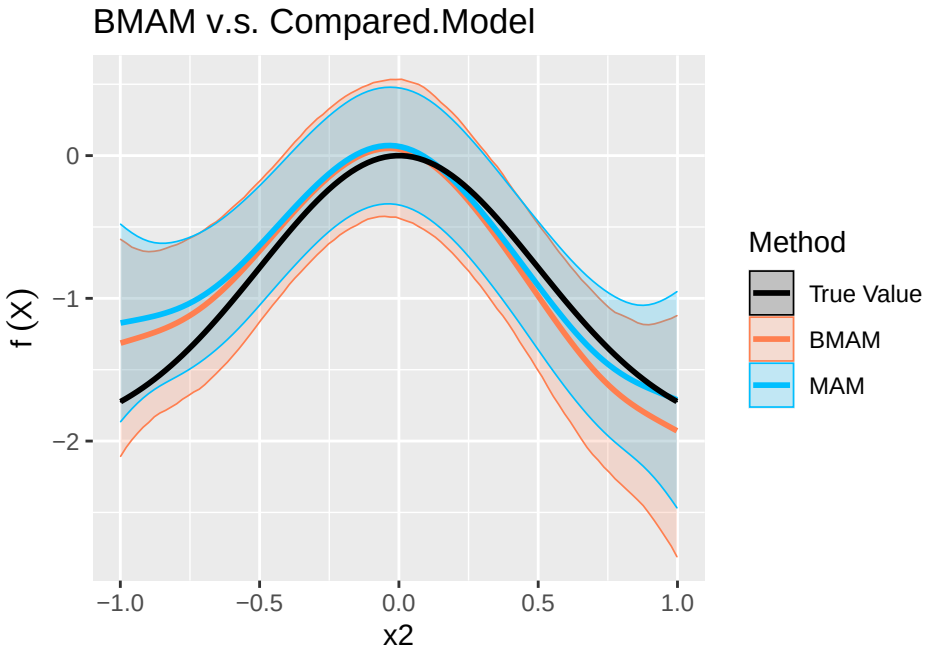
```
themam <- mam(smooth = list(s(x1),s(x2)),
                    re = y ~ (1+x3|id),
                    fe = ~ x3,
                    dat = dat,
                    margdat = dat,
                    preddat = bmam.fit$Preddat,
                    control = mam_control(
                      method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

```
plot(bmam.fit, compared.model = themam, smooth.function = simdata$f)
```




BMAM v.s. Conditional Model
f(X)
0
−1
−2
−3
−4
−1.0
−0.5
0.0
0.5
1.0
x2
Method
True Value
BMAM
Conditional Model

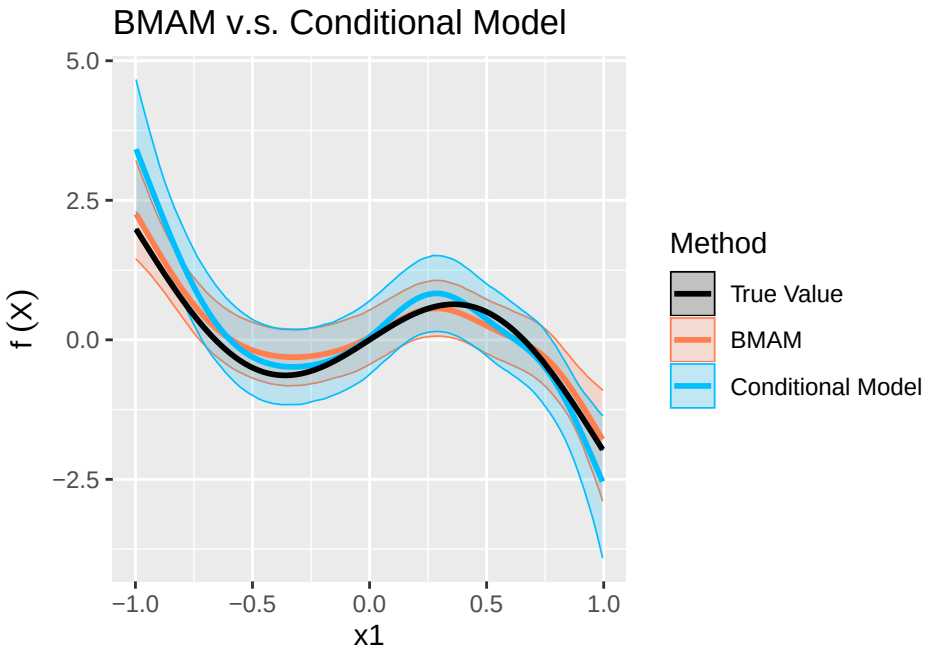

BMAM v.s. Compared.Model
f(X)
2
0
−2
−1.0
−0.5
0.0
0.5
1.0
x1
Method
True Value
BMAM
MAM

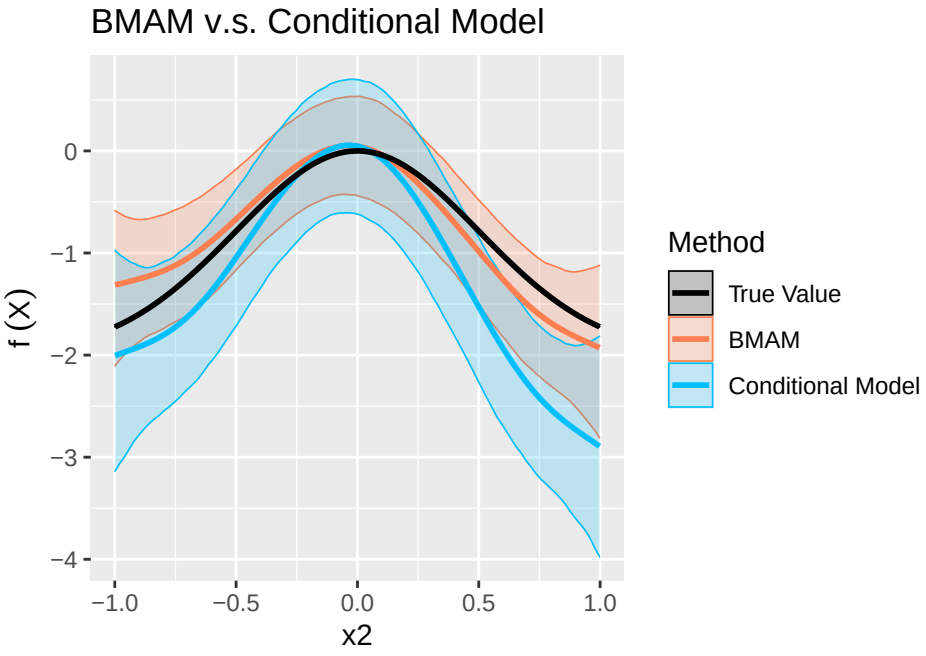


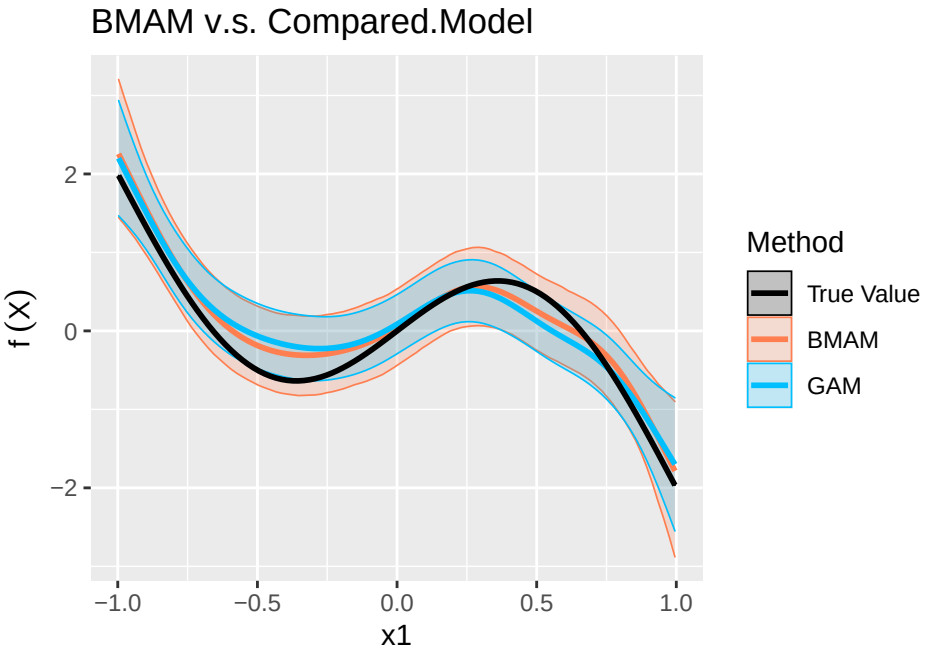
Generalized additive models We could also compare the the models with GAM. The `plot` function now supports the generalized additive models by `mgcv` package and Bayesian generalized additive models by `brms` package.

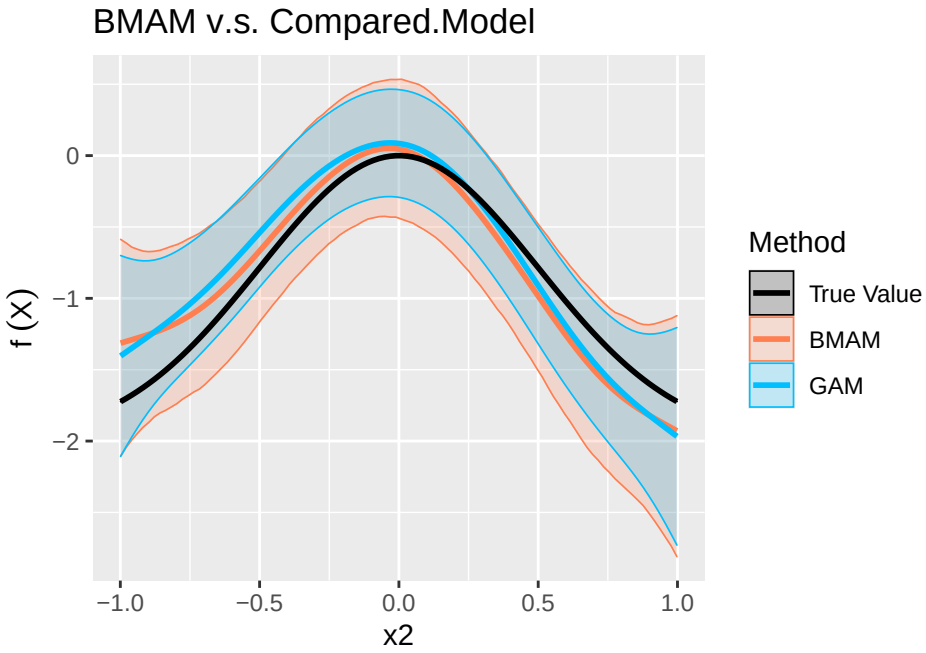
```
gam <- gam(y ~  x3 + s(x1) + s(x2),
           data=dat,family=binomial(),method="REML")
```

```
plot(bmam.fit, compared.model = gam, smooth.function = simdata$f)
```



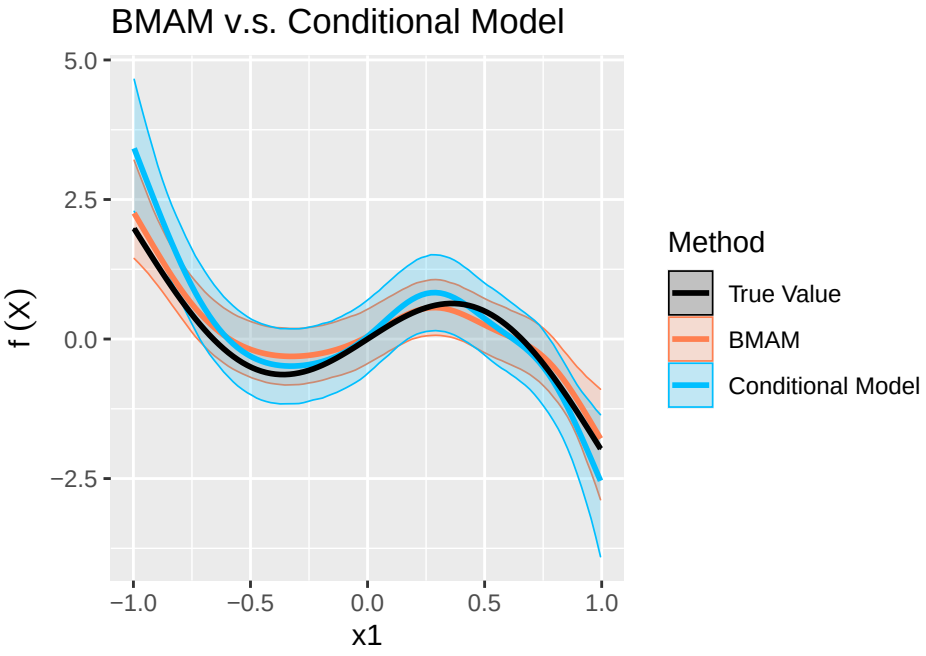

BMAM v.s. Conditional Model
0
−1
−2
−3
−4
f(X)
−1.0
−0.5
0.0
0.5
1.0
x2
Method
True Value
BMAM
Conditional Model

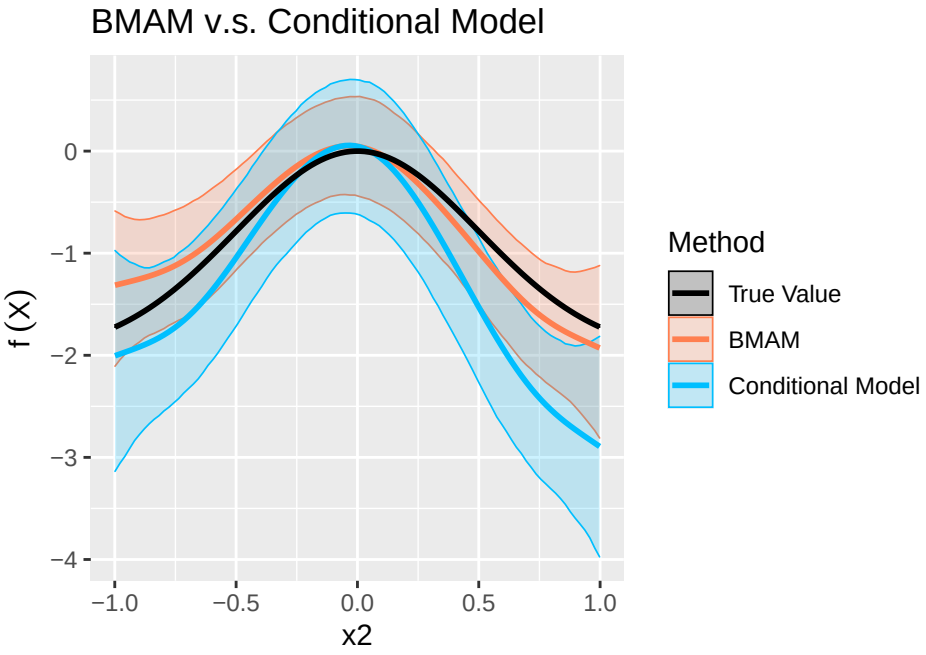

BMAM v.s. Compared.Model
2
0
−2
f(X)
−1.0
−0.5
0.0
0.5
1.0
x1
Method
True Value
BMAM
GAM

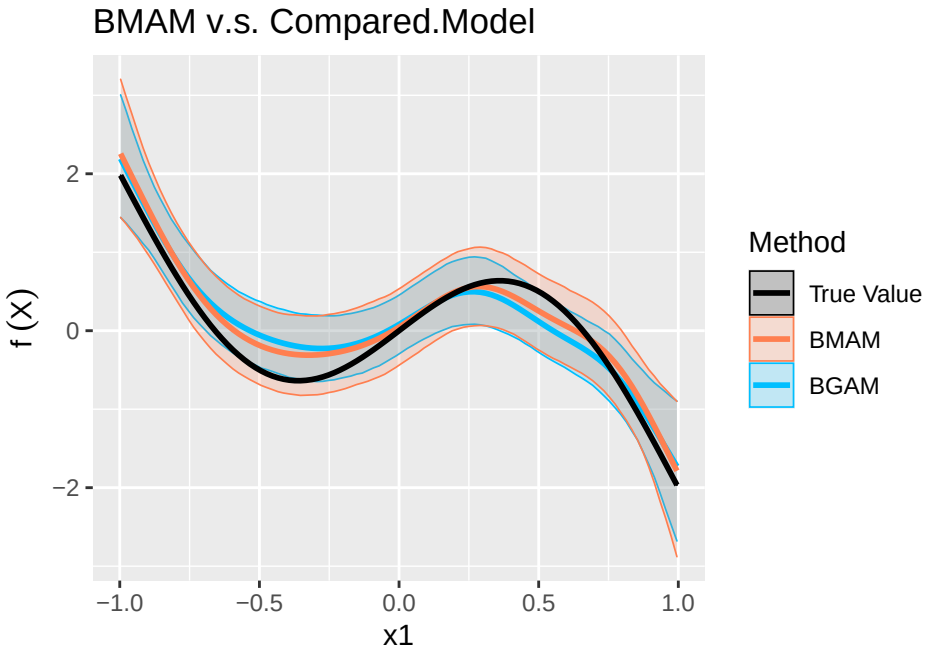


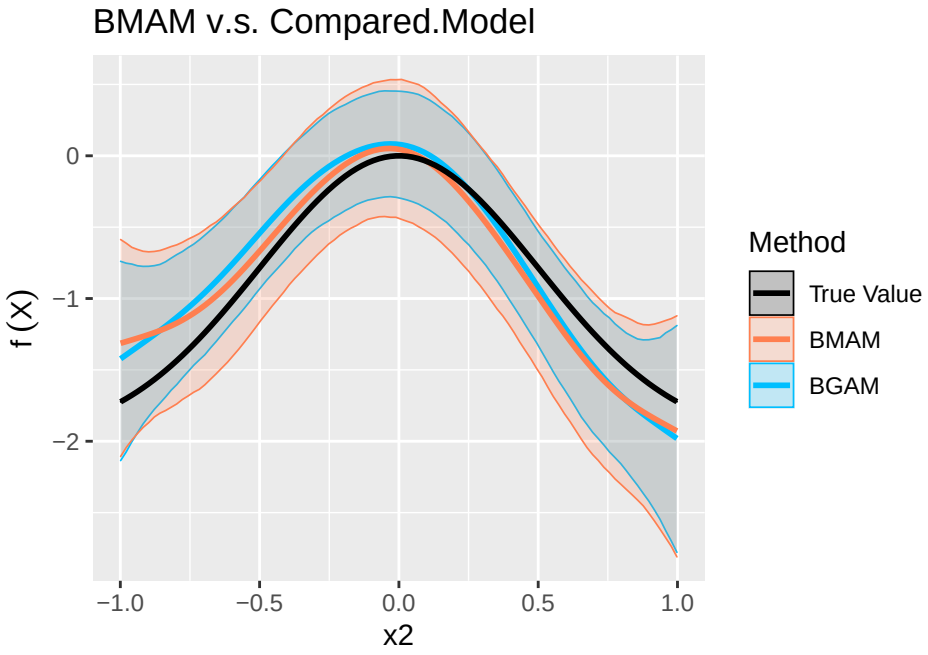

```
bgam <- brm(bf(y ~  x3 + s(x1) + s(x2)), data = dat,
            family = "bernoulli",
            cores = 4, seed = 4321, warmup = 1000,
            iter = 2000, chains = 4,
            refresh=0, backend = "cmdstanr")
#> Running MCMC with 4 parallel chains...
#>
#> Chain 2 finished in 20.6 seconds.
#> Chain 4 finished in 20.7 seconds.
#> Chain 1 finished in 22.0 seconds.
#> Chain 3 finished in 23.5 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 21.7 seconds.
#> Total execution time: 23.7 seconds.
```

```
plot(bmam.fit, compared.model = bgam, smooth.function = simdata$f)
```


BMAM v.s. Conditional Model
5.0
2.5
0.0
−2.5
f(X)
−1.0
−0.5
0.0
0.5
1.0
x1
Method
True Value
BMAM
Conditional Model


BMAM v.s. Conditional Model
0
−1
−2
−3
−4
f(X)
−1.0
−0.5
0.0
0.5
1.0
x2
Method
True Value
BMAM
Conditional Model





Center the smooth terms

In some cases, we may want to center the smooth functions.

We could add an argument `centered = TRUE` in `bmam` function, to estimate the centered smooth functions.

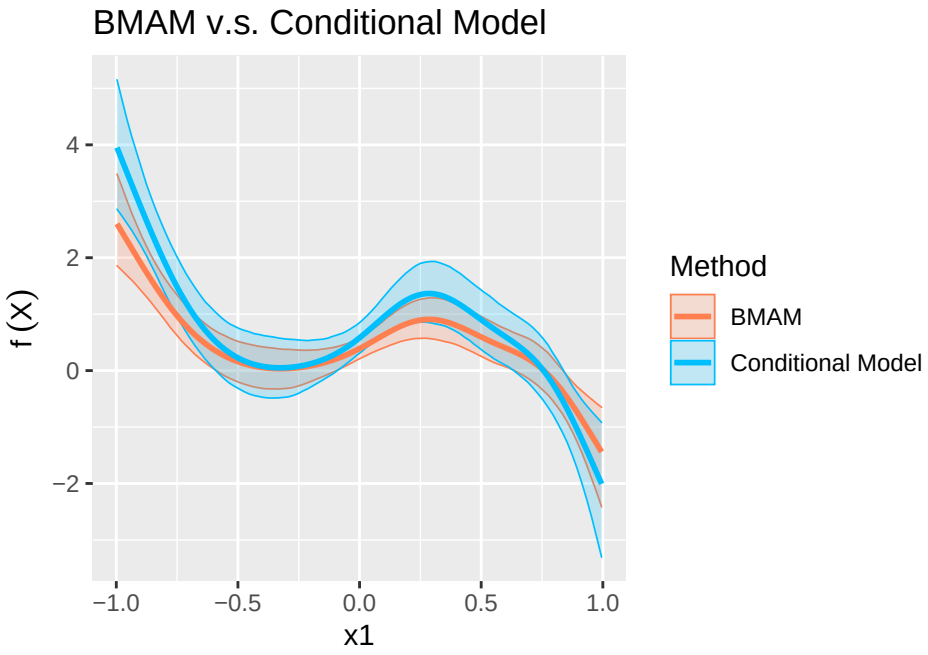
```
bmam.fit.centered <- bmam(object = model_brms, k=100, CIType="ETI", CI = 0.95, centered = TRUE)
```

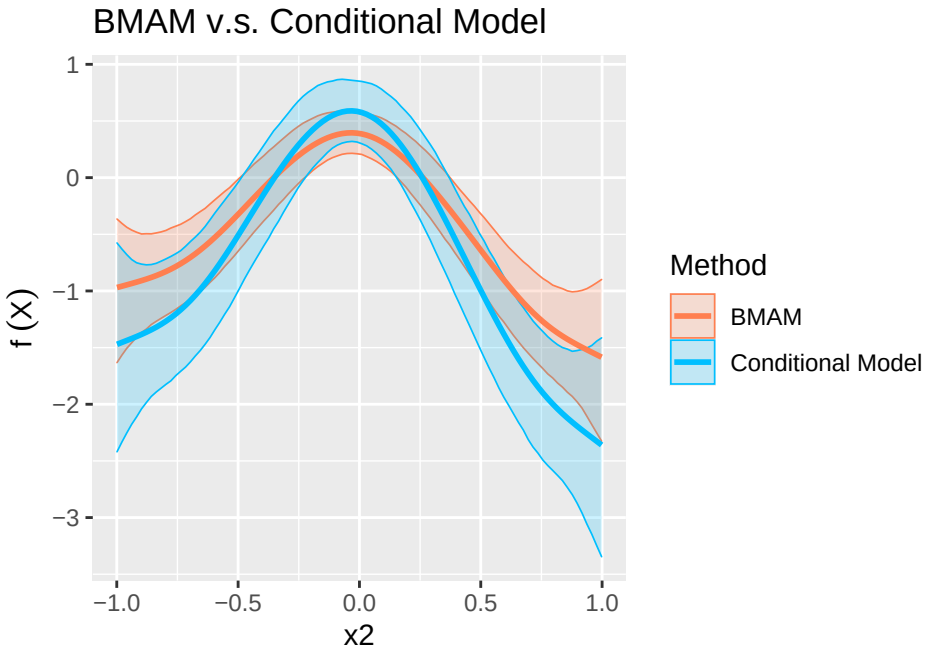
We can also fit a MAM with centered smooth functions,

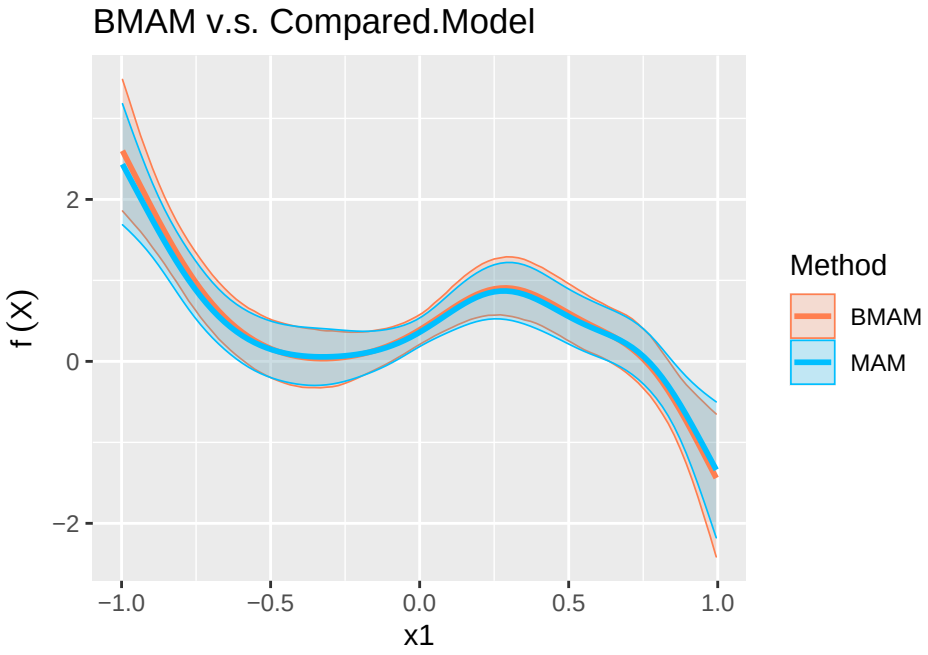
```
themam.centered <- mam(smooth = list(s(x1),s(x2)),
                       re = y ~ (1+x3|id),
                       fe = ~ x3,
                       dat = dat,
                       margdat = dat,
                       preddat = bmam.fit.centered$Preddat,
                       control = mam_control(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

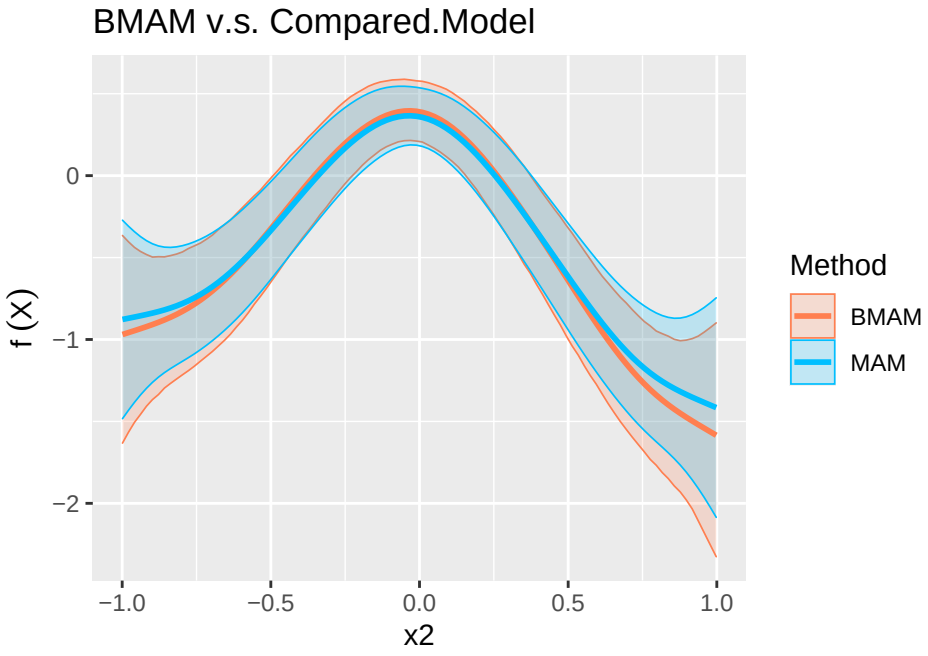
Then, we can draw the plots by `plot` function introduced above to compare the centered smooth terms in BMAM, the conditional model, and MAM.

```
plot(bmam.fit.centered, compared.model = themam.centered)
#> BMAM is centered.
```




BMAM v.s. Conditional Model
f (X)
1
0
−1
−2
−3
−1.0
−0.5
0.0
0.5
1.0
x2
Method
BMAM
Conditional Model


BMAM v.s. Compared.Model
f (X)
2
0
−2
−1.0
−0.5
0.0
0.5
1.0
x1
Method
BMAM
MAM



References

Bürkner, Paul-Christian. 2017. "Brms: An r Package for Bayesian Multilevel Models Using Stan." *Journal of Statistical Software* 80: 1–28.

Heagerty, Patrick J, and Scott L Zeger. 2000. "Marginalized Multilevel Models and Likelihood Inference." *Statistical Science* 15 (1): 1–26.

Hedeker, Donald, Stephen HC du Toit, Hakan Demirtas, and Robert D Gibbons. 2018. "A Note on Marginalization of Regression Parameters from Mixed Models of Binary Outcomes." *Biometrics* 74 (1): 354–61.

McGee, Glen, and Alex Stringer. 2022. "Flexible Marginal Models for Dependent Data." *arXiv Preprint arXiv:2204.07188*.

Wood, Simon N. 2006. *Generalized Additive Models: An Introduction with r*. chapman; hall/CRC.