# Bayesian regression methods

## Motivation

We are interested in using Bayesian regression techniques to characterize the nature of complex interactions. In addition to being an interesting statistical challenge, complex interactions are also relevant from both a mechanistic and public health point of view.

- mechanistic
- public health

Drawing from the mechanistic and public health contexts, there are three classes of interactions that are of interest.

- different types of interactions

However, non-additive interactions are difficult to detect.

First, there are different forms that an interaction can take on. For instance, in the linear regression setting, the form of the interaction must be explicitly specified; most commonly, this is done by multiplying two variables together. However,

Moreover, the number of possible interactions to consider quickly becomes intractable in high-dimensional settings. For instance, consider modelling 10 exposures in a linear regression setting. In order to be assessed, each interaction must be explicitly specified as a new term in the model. Including all the possible two-way interactions would involve adding $\binom{10}{2} = 45$ additional terms to the model, and all possible three-way interactions would add $\binom{10}{3} = 120$ additional terms.

- interactions are difficult to pick up
- consider the number of interactions that would have to be encoded explicitly in MLR

It is important also to acknowledge, here, that there is a limit to how many variables can be included in an interaction before it becomes incomprehensible to most humans.

## Bayesian kernel machine regression

Notation for kernel machine regression:

- $\mathbf{x}_m$ is a predictor variable in the predictor matrix $\mathbf{X}$ with $m = 1, \ldots, M$, measuring exposure variables or covariates in this case
- $\mathbf{x}_i$ is a vector of values for a single observation in $\mathbf{X}$ with $i = 1, \ldots, n$
- $x_{im}$ is an observation of $\mathbf{x}_m$
- $Y_i$ is an observation of $\mathbf{Y}$, measuring the health outcome in this case
- $h(\mathbf{x}_i)$ is the flexible function relating X to Y, represented by the kernel
- $k$ is the kernel function, the Gaussian in this case
- $\mathbf{K}$ is the $n \times n$ kernel matrix, with $(i, j)$th element $k(\mathbf{x}_i, \mathbf{x}_j)$
- $\rho$ is the tuning parameter inside the kernel function which controls smoothness
- $\tau$ is the tuning parameter multiplied by the kernel matrix to represent covariance between $h$ values
- $\boldsymbol{\epsilon}_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$ are the residuals of the response

BKMR specific notation:

- $r_m = 1/\rho_m$ is an augmented variable in $\mathbf{r}$ in the kernel matrix, for variable selection and controlling smoothness
- $\delta_m$ is an indicator variable in $\boldsymbol{\delta}$ which represents inclusion in the model
- $\mathcal{S}_g$ is a group of partitioned predictors with $g = 1, \ldots, G$
- $\{\delta_m | \mathbf{x}_m \in \mathcal{S}_g\}$ is an indicator variable in $\boldsymbol{\delta}_{\mathcal{S}_g}$ which represents inclusion of a parameter in group $g$ in the model
- $\pi$ is the prior probability of inclusion in the model
- $\lambda = \tau\sigma^{-2}$ is a convenient way to define the prior on $\tau$

**Kernel machine regression**

In this section, we introduce kernel machine regression, with attention to its specific implementation in BKMR. First proposed by Nadaraya [-@nadaraya_estimating_1964] and Watson [-@watson_smooth_1964], kernel machine regression is a nonparametric regression technique that can be used to capture nonlinear effects and nonadditive interactions. In this introduction, we follow the presentation of kernel machine regression provided by Bobb et al. [-@bobb_bayesian_2015]. To contextualize this method, we can start in the typical linear regression setting,

$$Y_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \epsilon_i,$$

where $Y_i$ measures a health outcome at a given point, $\mathbf{x}_i = (x_1, \ldots, x_M)^\top$ is a vector of M exposures or covariates (hereafter referred to as predictors), $\boldsymbol{\beta}$ is a vector of weights, and $\epsilon_i$ is a random variable from $\boldsymbol{\epsilon} \overset{\text{iid}}{\sim} N(0, \sigma^2)$. We can see that this function assumes that there is a linear relationship between the exposure and the response, and that the combined effects of multiple exposures are additive.

Kernel machine regression defines this relationship using a function $h : \mathbb{R}^M \to \mathbb{R}$, where

$$Y_i = h(\mathbf{x}_i) + \epsilon_i.$$

Here, $h(\cdot)$ is represented by the function $k(\cdot, \cdot)$, a kernel. The kernel controls the covariance, or the similarity, between values of $h(\mathbf{x})$ and as such ensures that points near each other on the prediction surface will have similar values — or, in other words, that the prediction surface will be smooth. In the case of kernel machine regression, we define a positive definite kernel where $k : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}$.

There are many choices of functions for $k$. BKMR uses the Gaussian kernel, also known as the radial basis function or, sometimes, the squared exponential kernel. The Gaussian kernel is defined as

$$k(\mathbf{x}, \mathbf{x}') = \exp\{-\frac{||\mathbf{x} - \mathbf{x}'||^2}{\rho}\},$$

where $||\mathbf{x} - \mathbf{x}'||^2 = \sum_{m=1}^{M} (x_m - x'_m)^2$ for a set of predictors values $\mathbf{x}$ and the predictor values of a second subject $\mathbf{x}'$. $\rho$ is a tuning parameter that controls the relationship between the correlation between two points and their distance. Greater values of $\rho$ will enforce more dependence between points and make the resulting function smoother. $h$ is related to $k$ by a multiplicative constant $\tau$, a tuning parameter which controls the vertical scale of $h$.

Now that we have defined $h$ and $k$, we can think about how to characterize the relationship between our response and predictors. Kernel machine regression is a nonparametric technique because it does not specify a functional form for this relationship. Hence, we will think about estimating the response at a particular query point. Operationally, Müller [-@muller_weighted_1987] demonstrated that kernel machine regression uses a weighted average of all the observations in the dataset to estimate the response, defined as

$$\bar{Y} = \frac{\sum_{i=1}^{n} w_i Y_i}{\sum_{i=1}^{n} w_i},$$

with some set of weights $\{w_i\}_{i=1}^n$. Intuitively, we want to weight the observations that are closer to the query point more heavily. Using the Gaussian kernel as a weight allows us to achieve this. Replacing the weight with the Gaussian kernel, we get

$$\bar{Y} = \frac{\sum_{i=1}^n k(\mathbf{x}, \mathbf{x}_i) Y_i}{\sum_{i=1}^n k(\mathbf{x}, \mathbf{x}_i)}.$$

As we move through the predictor space, we can think of the prediction as a continuous moving average of local points in the dataset. The correlation between two values of $h$ is defined as

$$\text{cor}(h_i, h_j) = \exp\{-\frac{\sum_{m=1}^M (x_{im} - x_{jm})^2}{\rho}\},$$

which allows us to see that values of $h$ near each other will have a higher correlation and thus similar values. This is also why the resulting function is smooth.

### Connection with mixed models

It is useful to make connections between this definition of kernel machine regression and mixed models. Liu et al. [-@liu_semiparametric_2007] demonstrated this by representing $h(\mathbf{x})$ as following a Gaussian process probability distribution,

$$h(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, \tau k(\mathbf{x}, \mathbf{x}')),$$

with mean function $m$ and covariance function $k$, where $\mathbf{x}$ is a vector of the predictor values, and $\mathbf{x}'$ contains the predictor values of another subject. A Gaussian process is a collection of random variables, of which any finite number follow a multivariate normal distribution. Here, we assume that the expected value of the $h$ function with input $\mathbf{x}$ is $\mathbf{0}$. We use $k$ for the covariance function, which represents the dependence between the function values with inputs $\mathbf{x}$ and $\mathbf{x}'$: $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(h(\mathbf{x}) - \mathbf{0})(h(\mathbf{x}') - \mathbf{0})]$.

Now, we can represent $h$ as a collection of variables from a Gaussian process. $h$ follows a multivariate normal distribution,

$$h(\mathbf{x}) \sim N(\mathbf{0}, \tau \mathbf{K}),$$

where $h(\mathbf{x}) = [h(\mathbf{x}_1), h(\mathbf{x}_2), \ldots, h(\mathbf{x}_n)]^\top$ and $\mathbf{K}$ is the kernel matrix. The kernel matrix is an $n \times n$ matrix with $(i, j)$th element $k(\mathbf{x}_i, \mathbf{x}_j)$. Now, returning back to the regression view, we can think of each $Y_i$ as following the distribution

$$Y_i \overset{\text{ind}}{\sim} N(h(\mathbf{x}_i), \sigma^2) \text{ for } i = 1, \ldots, n,$$

where $\sigma^2$ comes from the variance of the residuals.

### Toy example

In the following section, we introduce kernel machine regression with a toy example.

Consider the following case where we want to model the relationship between a single predictor and a response variable. Suppose the true relationship between $x$ and $Y$ is defined $Y = e^{\frac{x}{10}} + 2\sin(\frac{x}{2})$.

Figure @ref(fig:toy1) illustrates the shape of our simulated nonlinear data and the fit proposed by a simple linear regression. We can observe that the linear regression fails to capture the true nonlinear relationship.

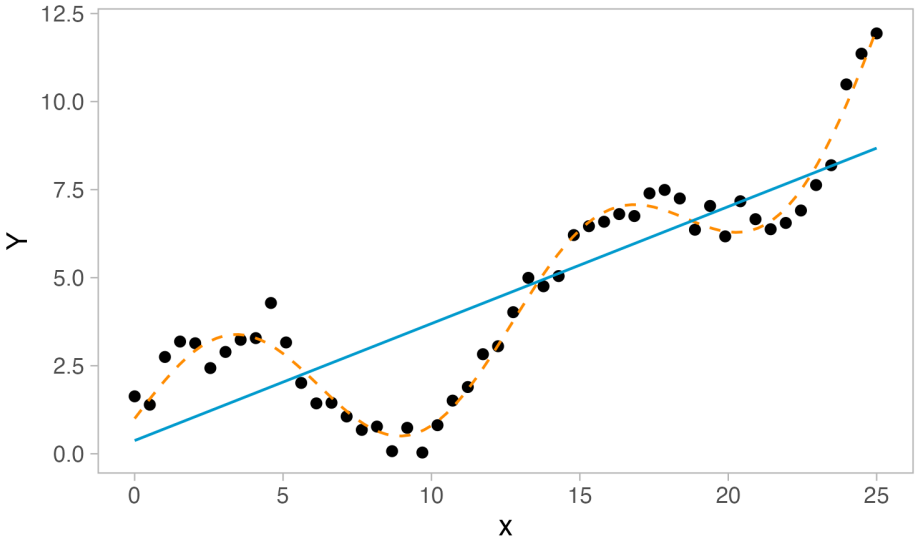


Figure 1: Nonlinear data with a true distribution (orange) and a fitted linear regression (blue).

In this case, this would lead to an underestimation of the true association between $x$ and $Y$. Now, we will try to capture this relationship using kernel machine regression.

To visualize how kernel machine regression works as a moving weighted average, we can consider a query point of 12.5.

Figure @ref(fig:toy2) identifies the query point and assigns corresponding weights to the neighboring points based on a normal distribution, which shares the same density as the Gaussian kernel. In this case, we will specify $\rho = 2$, which is synonymous with assigning the weights using a normal distribution with $\sigma^2 = 1$. We can see how an appropriate estimate for $h(12.5)$ can be obtained by taking a weighted average of the $x$'s, with those observations nearby weighted the most heavily.

Now, we fit a kernel machine regression on this data with $\rho = 2$ using the `stats` package in R [@r_core_team_r_2013].

We can see in Figure @ref(fig:toy3) that kernel machine regression captures the complex nonlinear relationship between $Y$ and $x$ and closely follows the true distribution. We do note, though, that the estimation is less precise at the tails, where there is less information provided by local observations. We can also use this example to consider the effect of various values of *rho* on the smoothness of the $h$ function.

### Variable selection

Now that we have defined kernel machine regression, we can extend it to the Bayesian paradigm. In this section, we discuss the two methods for Bayesian variable selection in BKMR: hierarchical variable selection and component-wise variable selection [@bobb_bayesian_2015].

Component-wise selection follows the same framework as variable selection in a typical Bayesian multiple regression except, instead of augmenting each predictor, we augment the kernel function as

$$k(\mathbf{x}, \mathbf{x}'|\mathbf{r}) = \exp\{-\sum_{m=1}^{M} r_m(x_m - x'_m)^2\},$$

where $\mathbf{r} = [r_1, \ldots, r_M]^\top$. We define $r_m = \frac{1}{\rho_m}$, the inverse of the tuning parameter $\rho_m$ for each $\mathbf{x}_m$, as we want to be able to shrink the influence of unimportant predictors in the model to $r_m = 0$. We now define

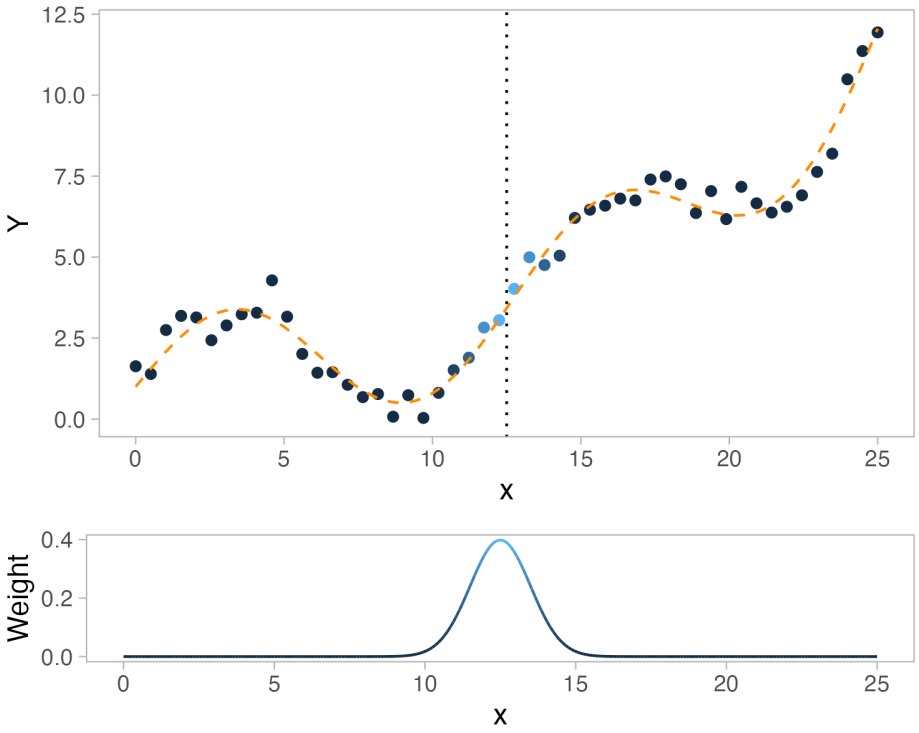


Figure 2: A query point of 12.5 and the weights of neighboring observations based on a Gaussian kernel

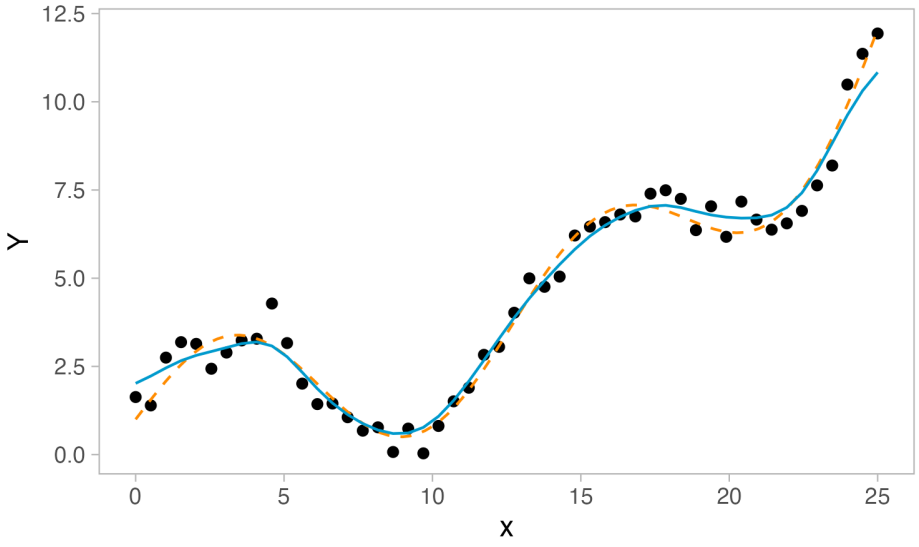


Figure 3: Fitted kernel machine regression with $\rho = 2$.

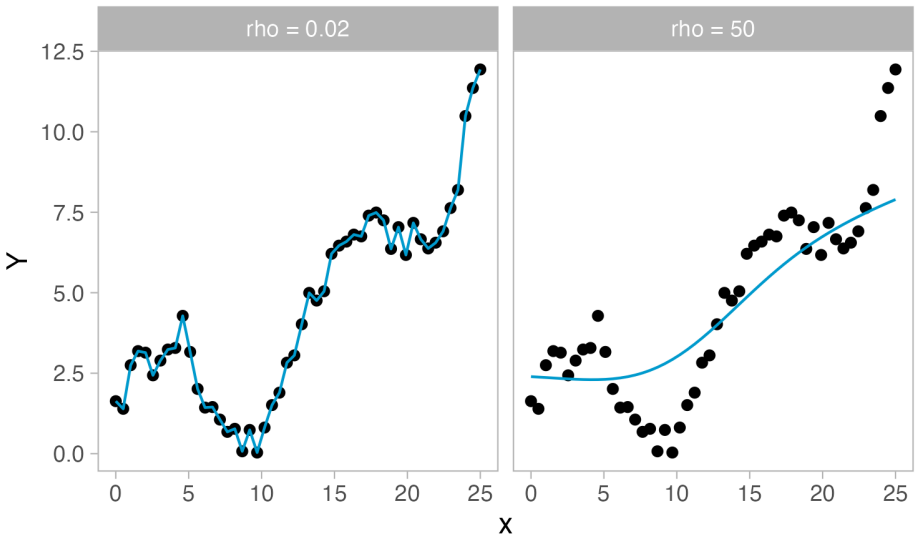


Figure 4: Fitted kernel machine regression.

the kernel matrix $\mathbf{K}_{\mathbf{X},\mathbf{r}}$ as the $n \times n$ matrix with $(i,j)$th element $k(\mathbf{x}, \mathbf{x}'|\mathbf{r})$. To allow $r_m$ to equal 0 with non-zero probability, we first define an indicator variable determining whether or not a predictor is included in the variable, which is distributed as

$$\delta_m \sim \text{Bernoulli}(\pi),$$

where $\pi$ is the prior probability of inclusion, and the posterior mean of $\delta_0$ is the posterior probability of inclusion of $\mathbf{x}_m$. Now, we can assume a "slab-and-spike" prior on $r_m$, distributed as

$$r_m|\delta_m \sim \delta_m f(\cdot) + (1-\delta_m)P_0,$$

where $f(\cdot)$ is some pdf with support $\mathbb{R}^+$ and $P_0$ denotes the density with point mass at 0.

While this process of component-wise variable selection works well in a typical multiple regression setting, it can lead to unreliable estimates in situations where the predictors are highly correlated with each other, which is common in exposure mixture studies. In this case, the correlated components contribute similar information to the model, and component-wise variable selection is not able to distinguish which predictor is important. BKMR deals with this problem by introducing hierarchical variable selection.

Hierarchical variable selection involves partitioning the predictors $\mathbf{x}_1, \ldots, \mathbf{x}_M$ a priori into groups $\mathcal{S}_g$ with $g = 1, \ldots, G$. These groups should be selected with the aim of keeping within-group correlation high and between-group correlation low. The indicators from $r_m|\delta_m$ are now distributed as

$$\boldsymbol{\delta}_{\mathcal{S}_g}|\omega_g \sim \text{Multinomial}(\omega_g, \boldsymbol{\pi}_{\mathcal{S}_g}), g = 1, \ldots, G, \omega_g \sim \text{Bernoulli}(\pi),$$

where $\boldsymbol{\delta}_{\mathcal{S}_g} = \{\delta_m | \mathbf{x}_m \in \mathcal{S}_g\}$ and $\boldsymbol{\pi}_{\mathcal{S}_g}$ are vectors of indicator variables and prior probabilities, respectively, of a predictor $\mathbf{x}_m$ in group $\mathcal{S}_g$ entering the model. By this approach, at most one predictor in each group is allowed to enter the model.

While hierarchical variable selection resolves the issue of multicollinearity, it requires specifying subgroups of predictors a priori and assumes that one predictor in each group can capture the information of the rest. Hence, care should be taken to justify the partitioning of predictors when taking this approach.

### Prior specification

In this section, we specify the prior distributions and parameters used in BKMR.

Can demonstrate understanding by discussing why this is a reasonable choice. For instance, the uniform on the inclusion parameter.

- $r_m|\delta_m \sim \delta_m \text{Unif}^{-1}(a_r, b_r) + (1 - \delta_m)P_0$
- $\rho_m = 1/r_m \sim \text{Unif}(a_r, b_r)$
- $\delta_m|\pi \sim \text{Bernoulli}(\pi)$
- $\delta_m|\pi \sim \text{Bernoulli}(\pi)$
- $\delta_m|\pi \sim \text{Bernoulli}(\pi)$
- $\pi \sim \text{Beta}(a_\pi, b_\pi)$
- $\sigma^{-2} \sim \text{Gamma}(a_\sigma, b_\sigma)$
- $\lambda = \tau\sigma^{-2} \sim \text{Gamma}(a_\lambda, b_\lambda)$

Discuss priors, what form do they take? what are the default settings?

### The MCMC algorithm

Briefly, we discuss the algorithm used in the BKMR package [@bobb_bayesian_2015; @bobb_statistical_2018], with commentary on its implications for the model fitting process.

BKMR uses a Markov chain Monte Carlo (MCMC) sampler with a mix Gibbs and Metropolis-Hastings samplers to estimate the posterior distributions. In particular, a Gibbs step is used to update the distribution of $\sigma^2$ while a Metropolis-Hastings step is used to update the distribution of $\lambda$. For component-wise and hierarchical variable selection, $(\mathbf{r}, \boldsymbol{\delta}, \boldsymbol{\omega})$ are sampled jointly using a Metropolis-Hastings sampling scheme.

While each distribution generated by the Gibbs step is always accepted, the distributions for $\lambda$ and $r_m$ generated by the Metroplis-Hastings steps are accepted based on an acceptance rate [@wagaman_probability_2021]. The standard deviation of the proposal distribution controls the acceptance rate and as such acts as a tuning parameter [@bobb_example_2017]. This standard deviation is a tuning parameter; in general, increasing the standard deviation leads to lower acceptance rates. Acceptance rates that are too low lead to slower convergence, but rates that are too high can cause convergence to an non-optimal distribution.

A major computational limitation of BKMR is that at each iteration of the MCMC algorithm, the $n \times n$ augmented kernel matrix $\mathbf{K}_{\mathbf{Z},\mathbf{r}}$ must be inverted multiple times. BKMR employs a Gaussian predictive process which involves specifying a set $l$ points, or "knots," that is a subset of the predictor space. The vector of predictors can be approximated by projection on this lower dimensional space, which allows the algorithm to perform inversions on an $l \times l$ matrix.

## Bayesian semiparametric regression

differences w/ bkmr

- makes distributional assumptions about the dataset
- kernel regression is computationally intensive w/ large datasets but can handle many predictors
- bsr highly dependent on the choice of function

### Connections to linear mixed model

### Toy example

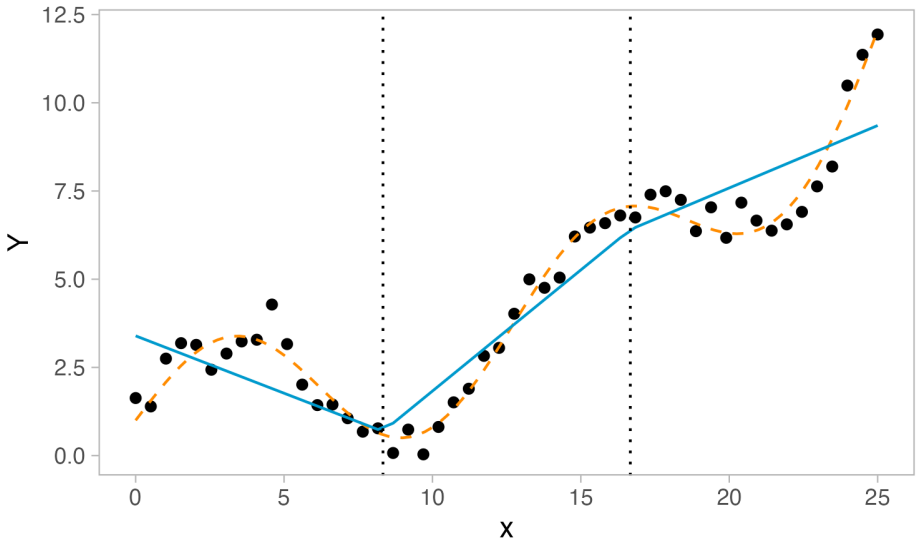


Figure 5: Linear spline regression

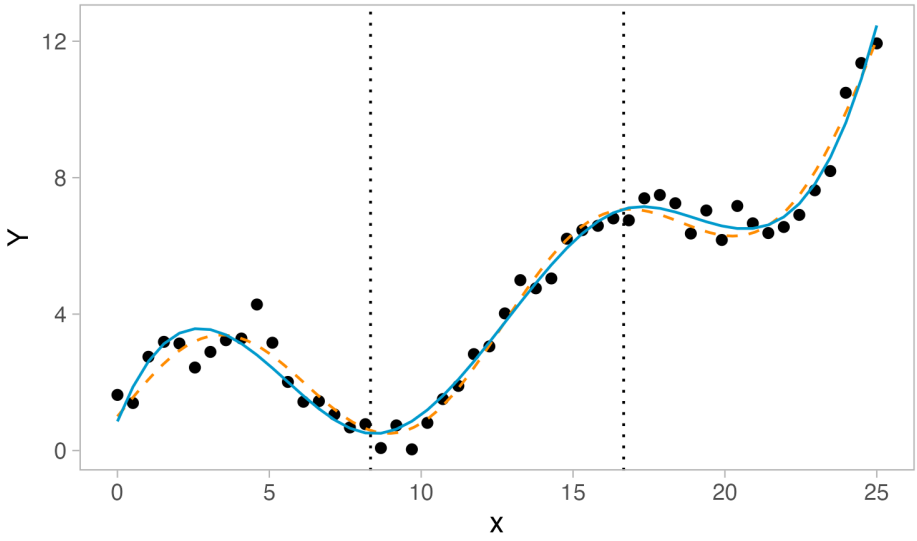


Figure 6: Spline regression

**Sparsity inducing priors**

**Prior specification**

**The MCMC algorithm**

## Detecting interactions

Discuss options for detecting interactions

BKMR can visualize relationship at various quantiles of other predictor, can conduct inference on inter-quantile difference

BSR can conduct inference

Potentially provide toy example