Multidimensional Monotonicity Discovery with mBART

Robert McCulloch

Collaborations with:
Hugh Chipman
Ed
Tom Shively

In Honor of Ed's 70th Birthday
December 10, 2021

Plan

- I) Review BART
- II) Introduce Monotone BART: mBART
- III) Monotonicity Discovery with mBART

Part I. BART (Bayesian Additive Regression Trees)

Data: n observations of y and $x = (x_1, ..., x_p)$

Suppose: $Y = f(x) + \epsilon, \quad \epsilon$ symmetric with mean 0

Bayesian Ensemble Idea: Approximate unknown $f(x)$ by the form

$$f(x) = g(x; \theta_1) + g(x; \theta_2) + ... + g(x; \theta_m)$$

$$\theta_1, \theta_2, \ldots, \theta_m \quad \text{iid} \sim \pi(\theta)$$

and use the posterior of f given y for inference.

BART: each $g(x; \theta_j)$ is a regression tree.

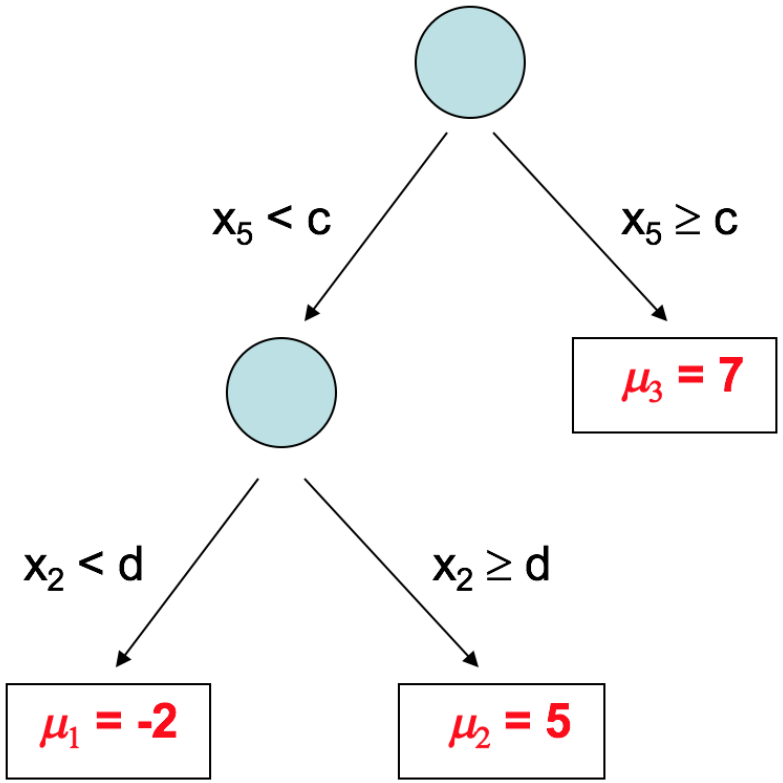
Key data calibration: Using y, set $\pi(\theta)$ so that $\text{Var}(f) \approx \text{Var}(y)$.

Beginning with a Single Tree Model

Let T denote the tree structure including the decision rules

Let $M = \{\mu_1, \mu_2, \ldots \mu_b\}$ denote the set of bottom node μ's.

Let $g(x; T, M)$ be a regression tree function that assigns a μ value to x



A single tree model:

$$Y = g(x; T, M) + \sigma z, \quad z \sim N(0,1)$$

Bayesian CART: Just add a prior $\pi(M, T)$

Bayesian CART Model Search
(Chipman, George, McCulloch 1998)

$$\pi(M, T) = \pi(M \mid T)\pi(T)$$

$\pi(T)$: Stochastic process to generate tree skeleton plus uniform prior on splitting variables and splitting rules.

$\pi(M \mid T) : (\mu_1, \mu_2, \ldots, \mu_b)' \sim N_b(0, \tau^2 I)$

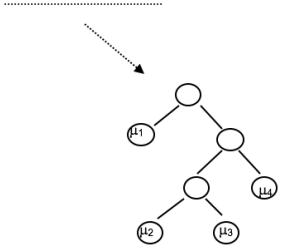
Closed form for $\pi(T \mid y)$ facilitates MCMC stochastic search for promising trees.

Moving on to BART

Bayesian Additive Regression Trees
(Chipman, George, McCulloch 2010)

The BART ensemble model

$$Y = g(x; T_1, M_1) + g(x; T_2, M_2) + \ldots + g(x; T_m, M_m) + \sigma z, \quad z \sim N(0, 1)$$



Each (T_i, M_i) identifies a single tree.

For each x, Y is the sum of m bottom node μ's, plus noise.

Number of trees m can be much larger than sample size n.

$g(x; T_1, M_1), g(x; T_2, M_2), ..., g(x; T_m, M_m)$ is a highly redundant "over-complete basis" with many many parameters.

Complete the Model with a "Regularization" Prior

$$\pi((T_1, M_1), (T_2, M_2), \ldots, (T_m, M_m), \sigma)$$

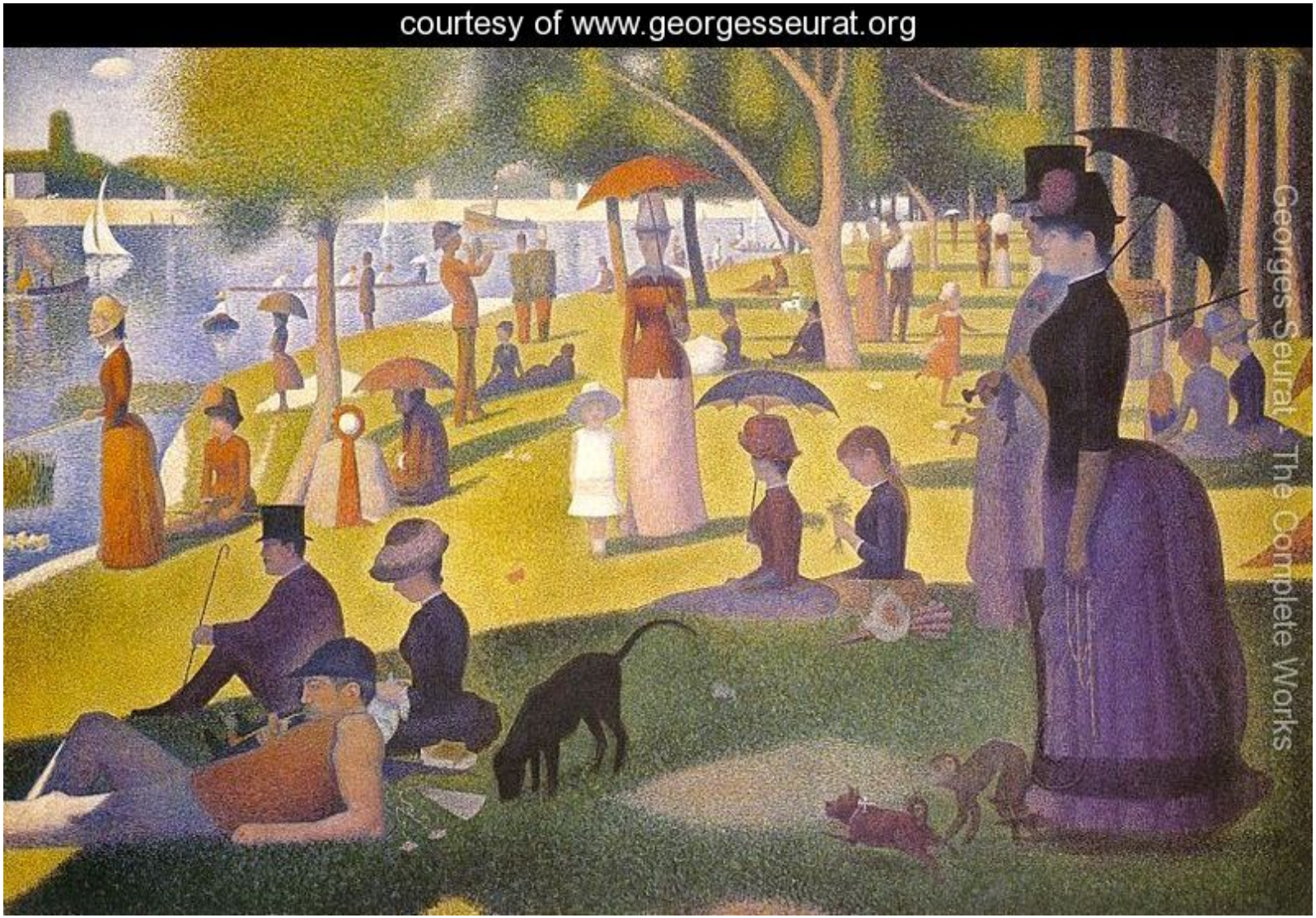
π applies the Bayesian CART prior to each (T_j, M_j) independently so that:

- Each T small.
- Each μ small.
- σ will be compatible with the observed variation of y.

The observed variation of y is used to guide hyperparameter settings for the μ and σ priors.

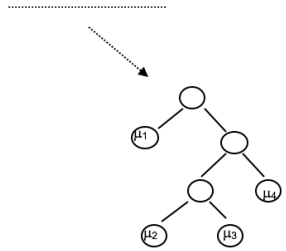
π keeps the contribution of each $g(x; T_i, M_i)$ small, to explain only a small portion of the fit.

Build up the fit, by adding up tiny bits of fit ..



Simple prior for a complex model !!!!!

$$Y = g(x; T_1, M_1) + g(x; T_2, M_2) + \ldots + g(x; T_m, M_m) + \sigma z, \quad z \sim N(0,1)$$



For each x, $f(x)$ is the sum of m bottom node μ's.

$$\mu \sim N(0, \tau^2), \textit{iid}.$$

$$\Rightarrow f(x) \sim N(0, m\tau^2), \quad \forall x.$$

When people try the R package it works just by doing
`res = BART::wbart(X,y)`.

Connections to Other Modeling Ideas

$$Y = g(x;T_1,M_1) + ... + g(x;T_m,M_m) + \sigma z$$
plus
$$\pi((T_1,M_1),....(T_m,M_m),\sigma)$$

Bayesian Nonparametrics:

- Lots of parameters (to make model flexible)
- A strong prior to shrink towards simple structure (regularization)
- BART shrinks towards additive models with some interaction

Boosting:

- Fit becomes the cumulative effort of many weak learners

Dynamic Random Basis Elements:

- $g(x; T_1, M_1), ..., g(x; T_m, M_m)$ are dimensionally adaptive

A Sketch of the BART MCMC Algorithm

$$Y = g(x;T_1,M_1) + \ldots + g(x;T_m,M_m) + \sigma z$$
plus
$$\pi((T_1,M_1),\ldots.(T_m,M_m),\sigma)$$

Bayesian Backfitting: Outer loop is a "simple" Gibbs sampler

$$\begin{aligned} (T_i, M_i) &\mid Y, \text{ all other } (T_j, M_j), \text{ and } \sigma \\ \sigma &\mid Y, (T_1, M_1, \ldots, \ldots, T_m, M_m) \end{aligned}$$

To draw (T_i, M_i) above, subtract the contributions of the other trees from both sides to get a simple one-tree model.

We integrate out M to draw T and then draw $M \mid T$.

... as the MCMC runs, trees in the sum will grow and shrink, swapping fit amongst them ....

Using the MCMC Output to Draw Inference

Each iteration d results in a draw from the posterior of f

$$\hat{f}_d(\cdot) = g(\cdot; T_{1d}, M_{1d}) + \cdots + g(\cdot; T_{md}, M_{md})$$

To estimate $f(x)$ we simply average the $\hat{f}_d(\cdot)$ draws at x

Posterior uncertainty is captured by variation of the $\hat{f}_d(x)$
eg, 95% credible region estimated by middle 95% of values

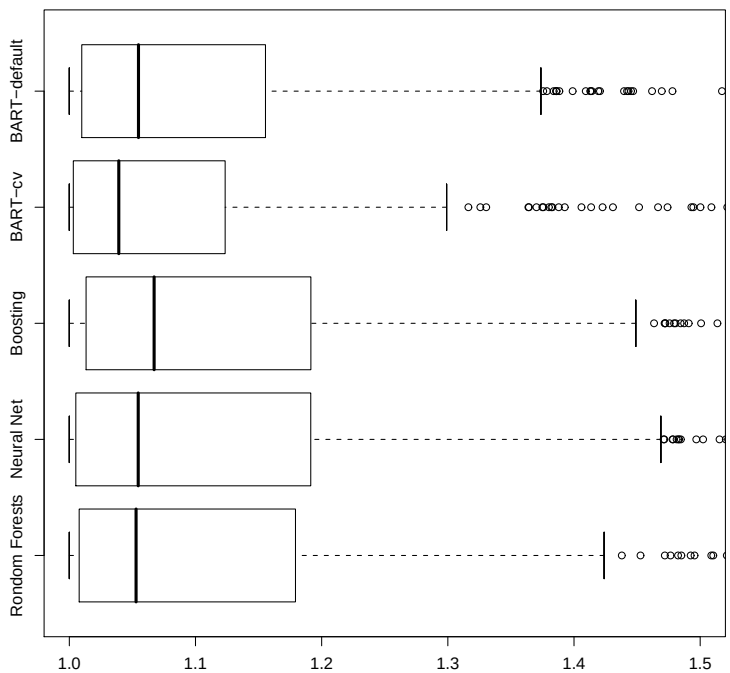
Out of Sample Prediction

Predictive comparisons on 42 data sets.

Data from Kim, Loh, Shih and Chaudhuri (2006) (thanks Wei-Yin Loh!)

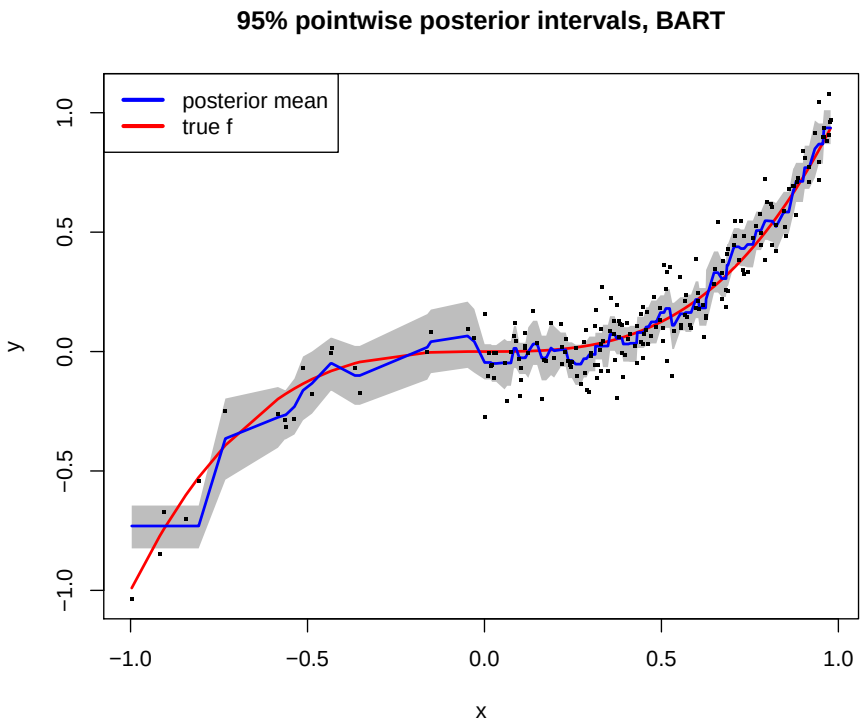
- $p = 3$ to 65, $n = 100$ to 7,000.
- for each data set 20 random splits into 5/6 train and 1/6 test
- use 5-fold cross-validation on train to pick hyperparameters (except BART-default!)
- gives 20*42 = 840 **out-of-sample predictions**, for each prediction, divide rmse of different methods by the smallest

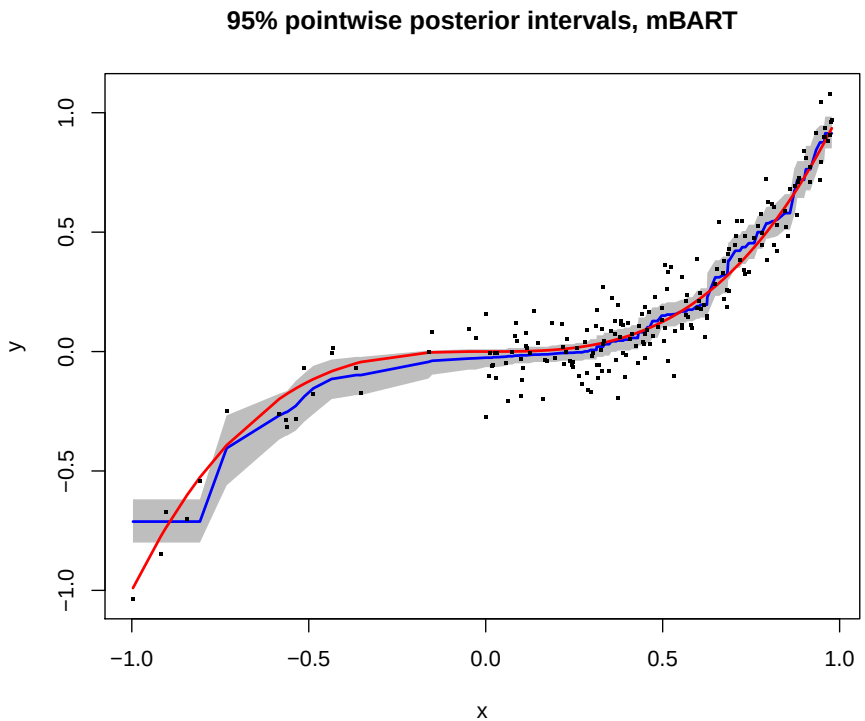
+ each boxplots represents 840 predictions for a method
+ 1.2 means you are 20% worse than the best
+ BART-cv best
+ BART-default (use default prior) does amazingly well!!



Automatic Uncertainty Quantification

A simple simulated 1-dimensional example





Note: mBART on the right plot to be introduced next

Part II. Monotone BART - mBART

mBART: Multidimensional Monotone BART
(Chipman, George, McCulloch, Shively 2021)

Key Idea:

Approximate multivariate monotone functions by the sum of many single monotonic tree models.

This works because

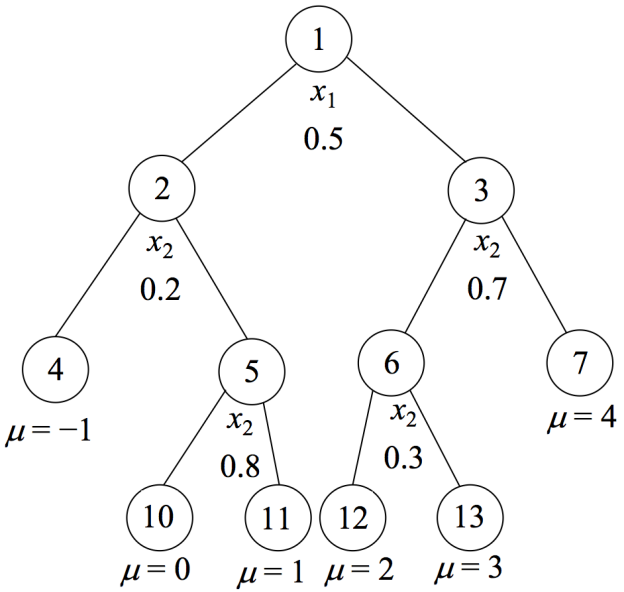
1. We can easily define a notion of "monotonic" for a single tree.
2. Because trees are simple, we can construct an MCMC which respects the constraints.

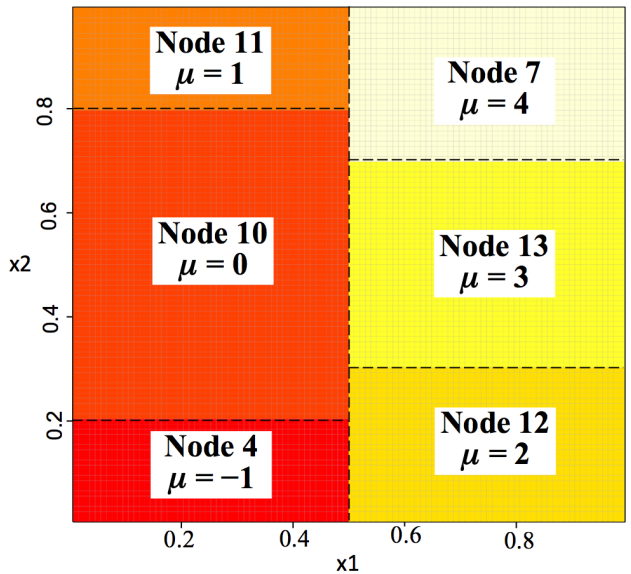
So,

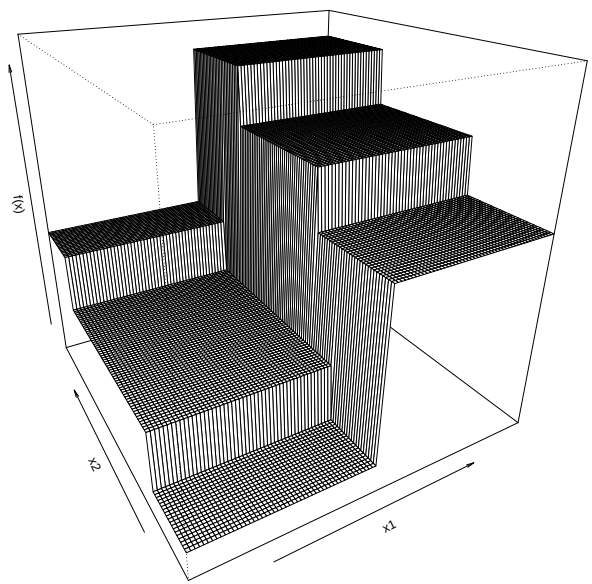
we can still use the BART approach

but now complex montonic functions are built as the sum of many single tree models, each of which is monotonic.

An Example of a Monotonic Tree

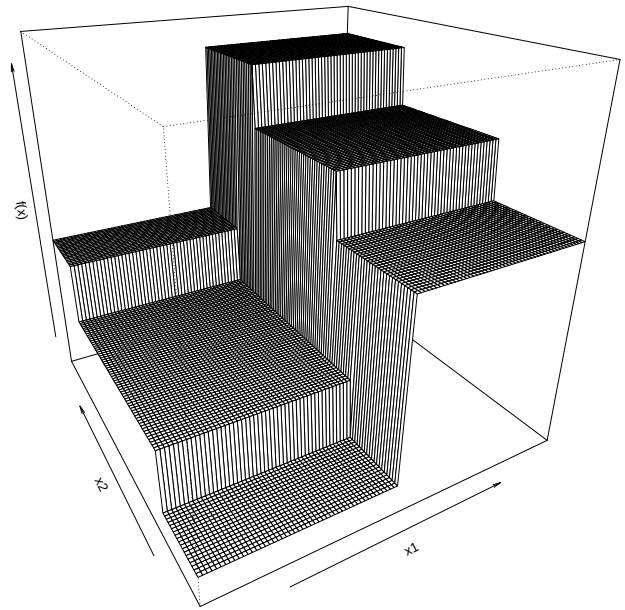






Three different views of a bivariate monotonic tree.

In what sense is this tree monotonic?

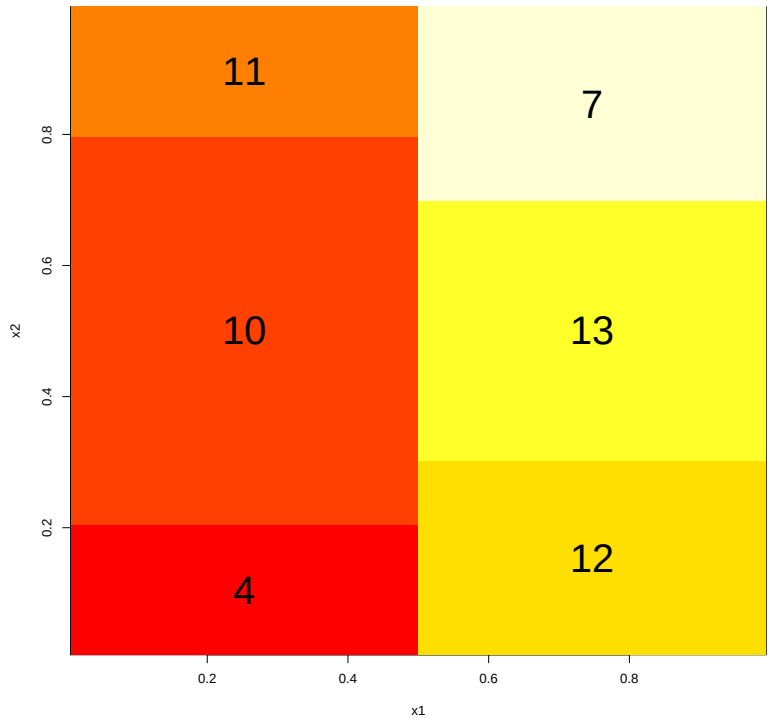


A function g is said to be *monotonic* in x_i if for any $\delta > 0$,

$$g(x_1, x_2, \ldots, x_i + \delta, x_{i+1}, \ldots, x_k; T, M) \geq g(x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_k; T, M).$$

For simplicity and wlog, let's restrict attention to monotone nondecreasing functions.

Constraining a tree to be monotone is easy: we simply constrain the mean level of a node to be greater than those of its "below-neighbors", and less than those of its "above-neighbors".



The mean level of node 13 must be greater than those of 10 and 12 and less than that of node 7.

The mBART Prior

Recall the BART parameter

$$\theta = ((T_1, M_1), (T_2, M_2), \ldots, (T_m, M_m), \sigma)$$

Let $S = \{\theta : \text{every tree is monotonic in a desired subset of } x_i's\}$

To impose the monotonicity we simply truncate the BART prior $\pi(\theta)$ to the set S

$$\pi^*(\theta) \propto \pi(\theta)\, I_S(\theta)$$

where $I_S(\theta)$ is 1 if *every* tree in θ is monotonic.

A New BART MCMC "Christmas Tree" Algorithm

$$\pi((T_1, M_1), (T_2, M_2), \ldots, (T_m, M_m), \sigma \mid y))$$

Bayesian Backfitting again: Iteratively sample each (T_j, M_j) given (y, σ) and other (T_j, M_j)'s

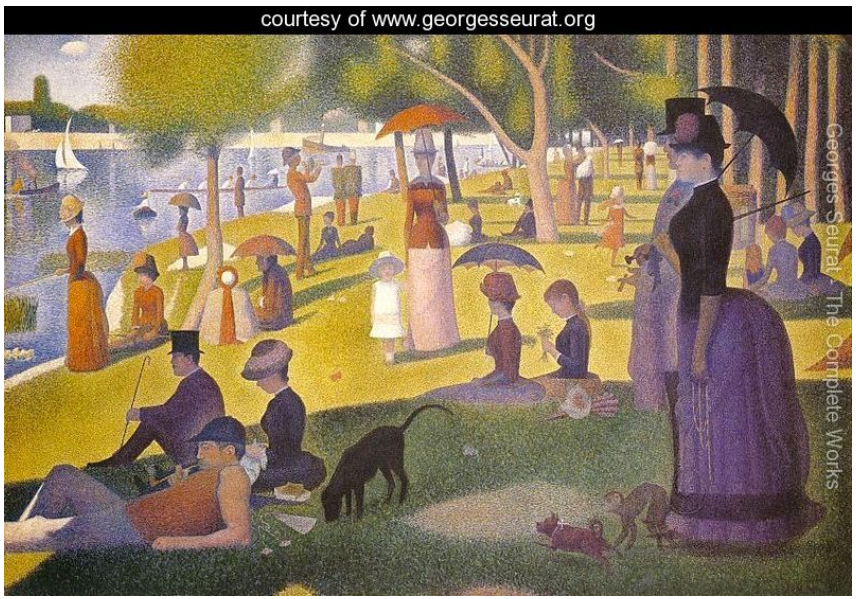
Each $(T^0, M^0) \to (T^1, M^1)$ update is sampled as follows:

- Denote move as $(T^0, M^0_{Common}, M^0_{Old}) \to (T^1, M^0_{Common}, M^1_{New})$
- Propose T^* via birth, death, etc.
- If M-H with $\pi(T, M \mid y)$ accepts (T^*, M^0_{Common})
  - Set $(T^1, M^1_{Common}) = (T^*, M^0_{Common})$
  - Sample M^1_{New} from $\pi(M_{New} \mid T^1, M^1_{Common}, y)$

Only $M^0_{Old} \to M^1_{New}$ needs to be updated.

Works for both BART and mBART.

the joy of working with Ed !!!!!!



There is the math junk and the code junk.

But what is it like working with Ed???

```
//--------------------
// how does node n relate to this
//'a': neighbor above, 'b': neighbor below, 'd': disjoint, 'x': nothing
char tree::nhb(tree_p n, xinfo& xi, std::vector<int>& vc)
//note:
//  each node is a region of the form \cap [Lv,Uv] v=0,1,..,(p-1)
{
   if(this==n) return 's'; //s means self
   size_t p = xi.size(); //need to loop over the p variables

   int mL,mU,oL,oU; //my range, other range
   short nind=0;

   //first loop over all variables to see if n is disjoint from this
   for(size_t v=0;v<p;v++) {
      mL=0; oL=0;
      mU = oU = xi[v].size()-1;
      rg(v,&mL,&mU);
      n->rg(v,&oL,&oU);
      //a neighbor will be 2 away since [L,U] are the usable ones,
         //for example: 'b': [oL,oU] used cut point=C, [mL,mU]
         //would have oU,C,mL in sequence
      if(oU < mL-2 || oL > mU+2 ) return 'd';
   }

   // now loop over all variables in vc to see if n is above or below this
   for(size_t i=0;i<vc.size();i++) {
      v = vc[i];
      mL=0; oL=0;
      mU = oU = xi[v].size()-1;
      rg(v,&mL,&mU);
      n->rg(v,&oL,&oU);
      //a neighbor will be 2 away since [L,U] are the usable ones,
```

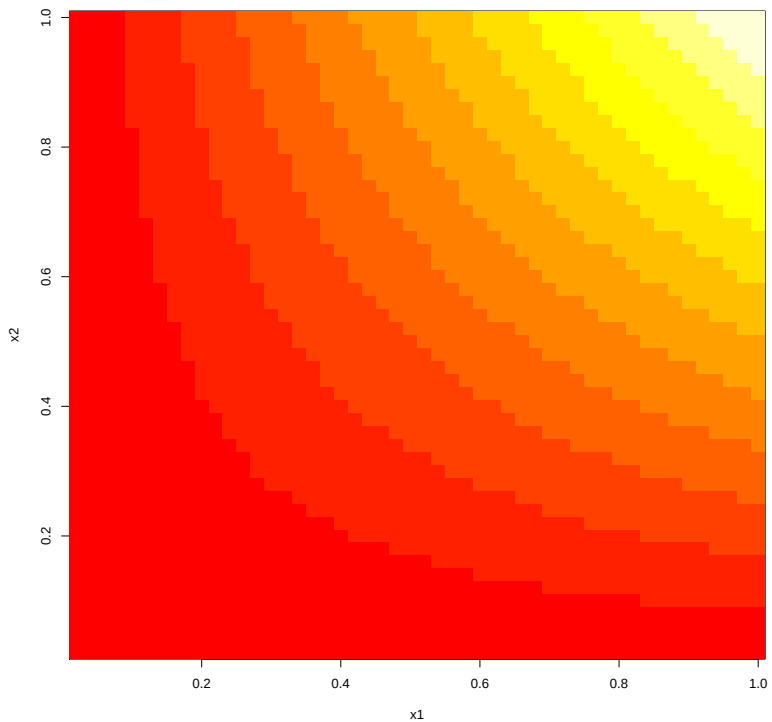
It's fun !!!!!!!!!!!!!!!!!!!!!!!!!!

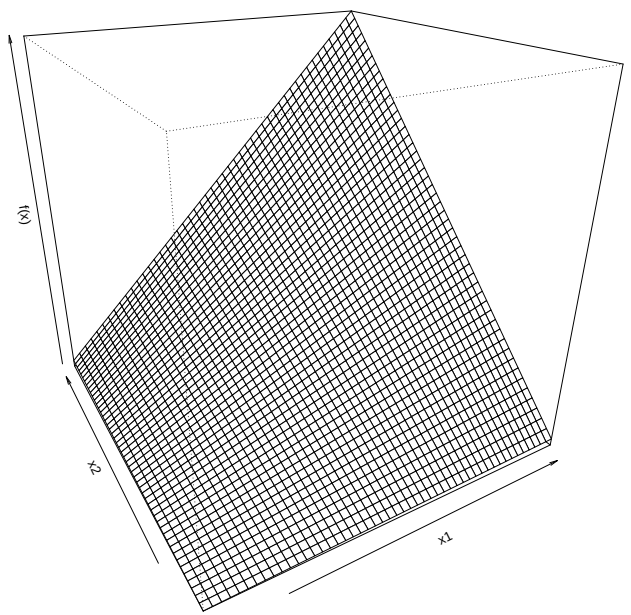
Example: Product of two x's

Let's consider a very simple simulated monotone example:

$$Y = x_1\, x_2 + \epsilon, \quad x_i \sim \text{Uniform}(0, 1).$$

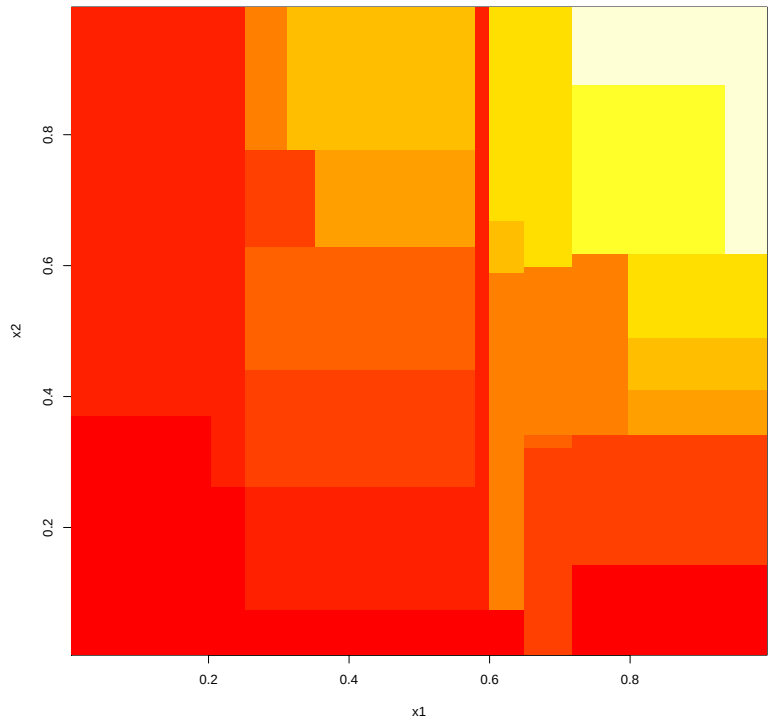
Here is the plot of the true function $f(x_1, x_2) = x_1\, x_2$

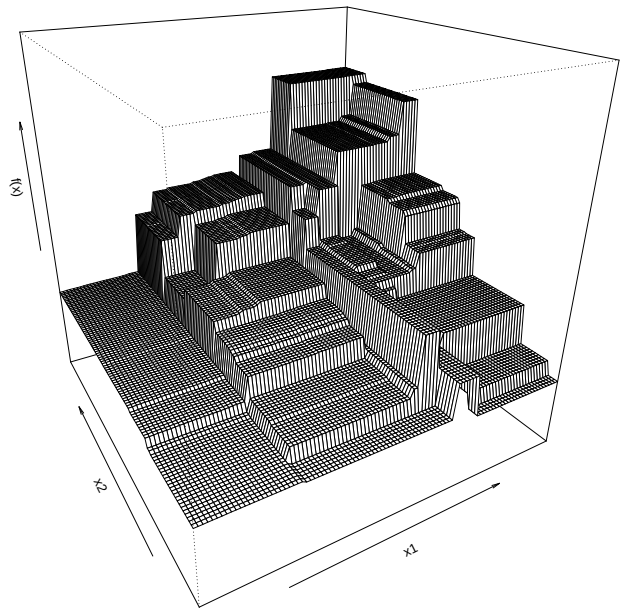




First we try a single (just one tree), unconstrained tree model.

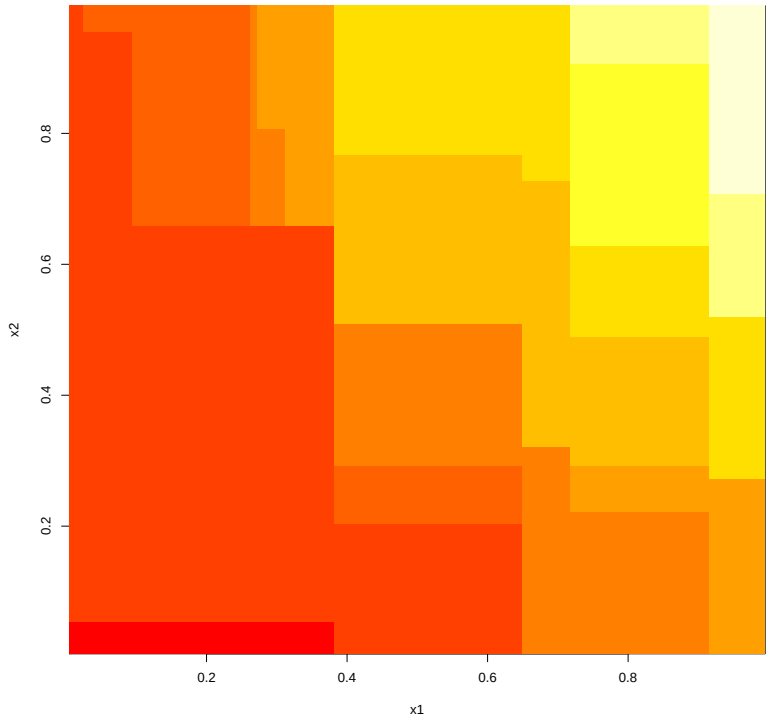
Here is the graph of the fit.

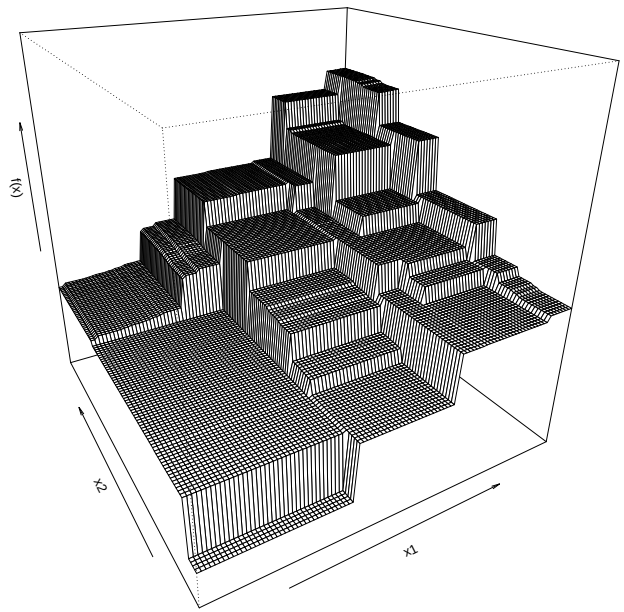




The fit is not terrible, but there are some aspects of the fit which violate monotonicity.

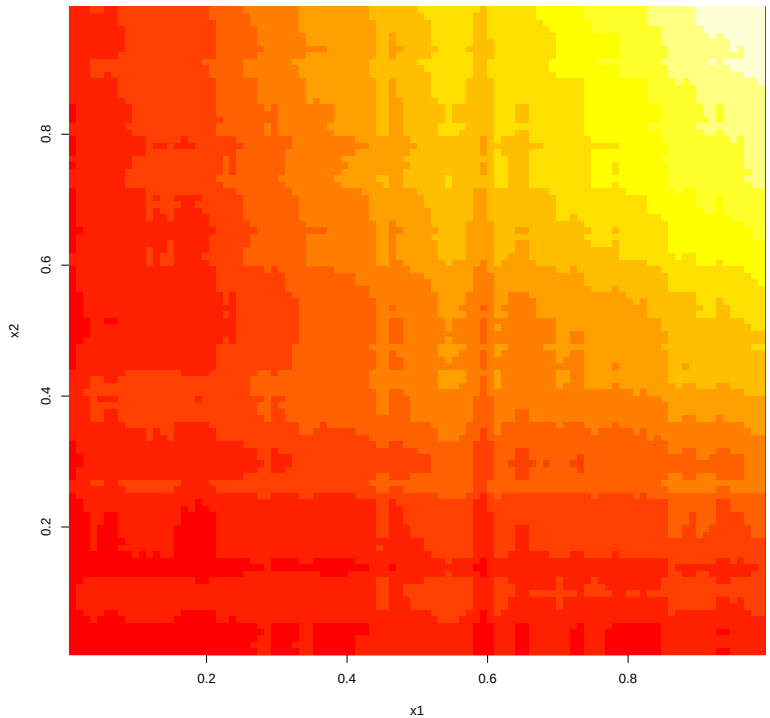
Here is the graph of the fit with the monotone constraint:

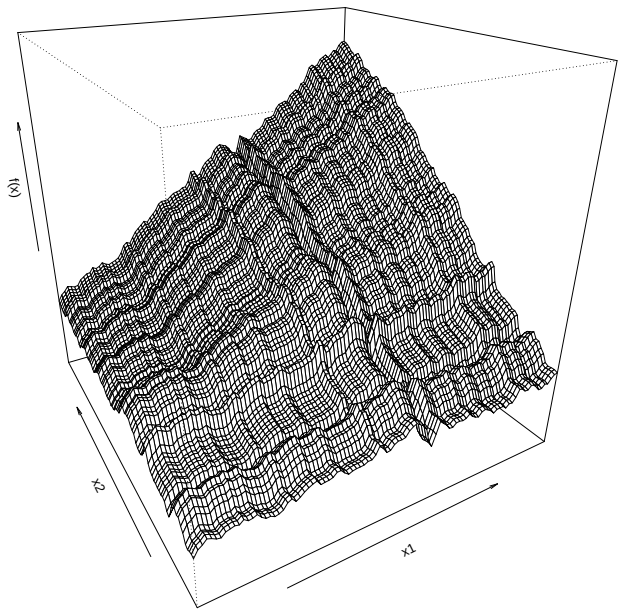




We see that our fit is monotonic, and more representative of the true f.

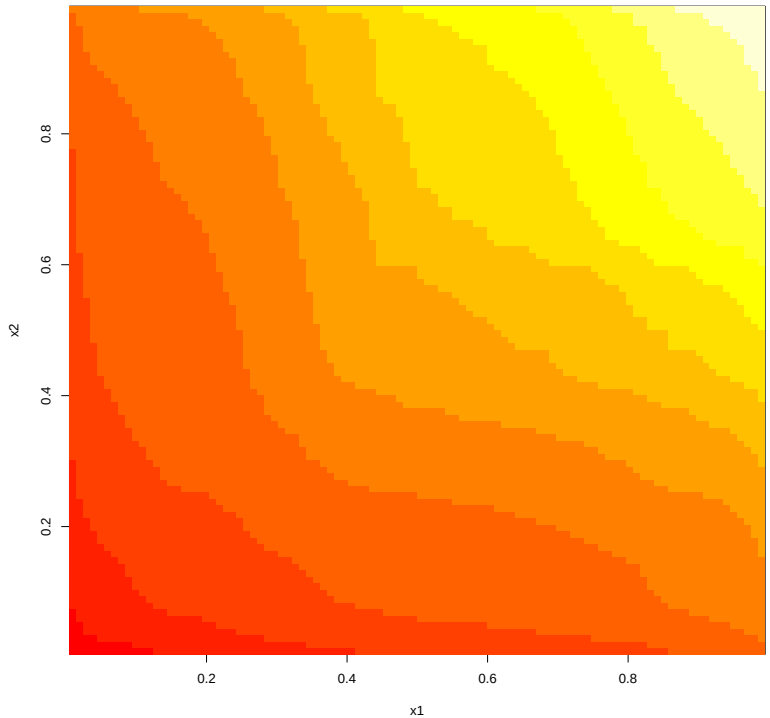
Here is the unconstrained BART fit:

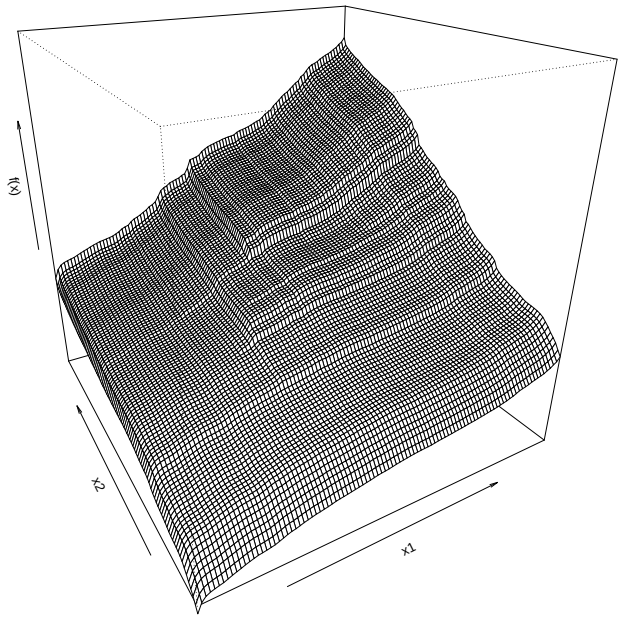




Much better (of course) but not monotone!

And, finally, the constrained BART fit:





Not Bad!

Same method works with any number of x's!

Example: MSE Reduction by Monotone Regularization

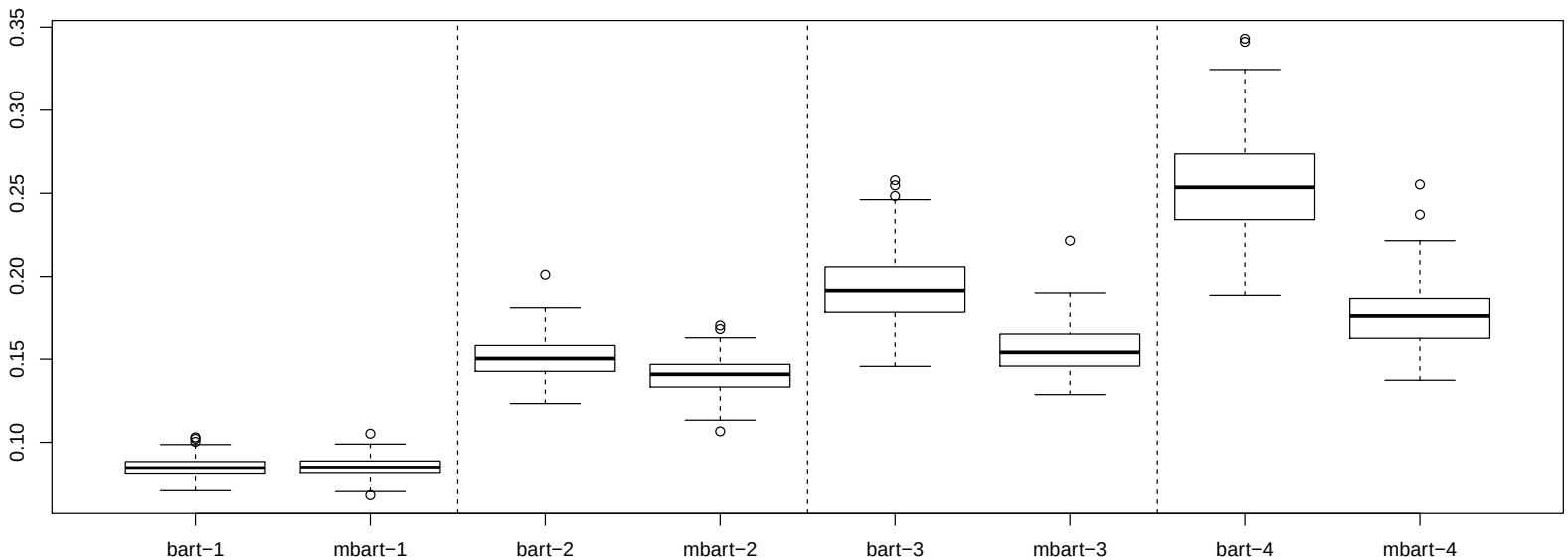
$$Y = x_1\, x_2^2 + x_3\, x_4^3 + x_5 + \epsilon,$$

$$\epsilon \sim N(0, \sigma^2), \quad x_i \sim \text{Uniform}(0, 1).$$

For various values of σ, we simulated 5,000 observations.

RMSE improvement over unconstrained BART

σ	Monotone BART RMSE	Unconstrained BART RMSE	Percentage Increase
0.5	0.14	0.16	14%
1.0	0.17	0.28	65%



$$\sigma = 0.2, 0.5, 0.7, 1.0$$

Part III. Discovering Monotonicity with mBART

Suppose we don't know if $f(x)$ is monotone up, monotone down or even monotone at all.

Of course, a simple strategy would be simply compare the fits from BART and mBART.

Good news, we can do much better than this!

As we'll now see, mBART can be deployed to simultaneously estimate all the monotone components of f.

With this strategy, monotonicity can be discovered rather than imposed!

The Monotone Decomposition of a Function

To begin simply, suppose x is one-dimensional and f is of bounded variation.

Any such f can be uniquely written (up to an additive constant) as the sum of a monotone up function and a monotone down function

$$f(x) = f_{up}(x) + f_{down}(x)$$

where

- *when $f(x)$ is increasing, $f_{up}(x)$ increases at the same rate and is flat otherwise,*
- *when $f(x)$ is decreasing, $f_{down}(x)$ decreases at the same rate and is flat otherwise.*

More precisely, when f is differentiable,

$$f'_{up}(x) = \begin{cases} f'(x) & \textit{when } f'(x) > 0 \\ 0 & \textit{when } f'(x) \leq 0 \end{cases}$$

and

$$f'_{down}(x) = \begin{cases} f'(x) & \textit{when } f'(x) < 0 \\ 0 & \textit{when } f'(x) \geq 0 \end{cases}$$

Notice the *orthogonal decomposition* of f'

$$f'(x) = f'_{up}(x) + f'_{down}(x)$$

The Monontone Discovery Strategy with mBART

Key Idea: To discover the monotone decomposition of f, we embed $f(x)$ as a two-dimensional function in R^2,

$$f(x) = f^*(x, x) = f_{up}(x) + f_{down}(x).$$

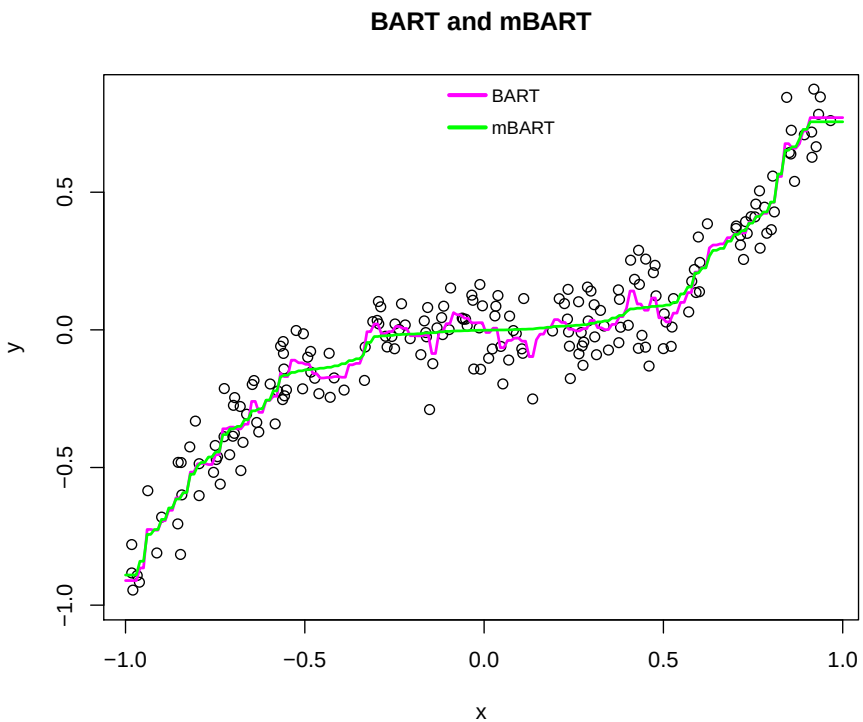
Letting $x_1 = x_2 = x$ be duplicate copies of x, we apply mBART to estimate $f^*(x_1, x_2)$

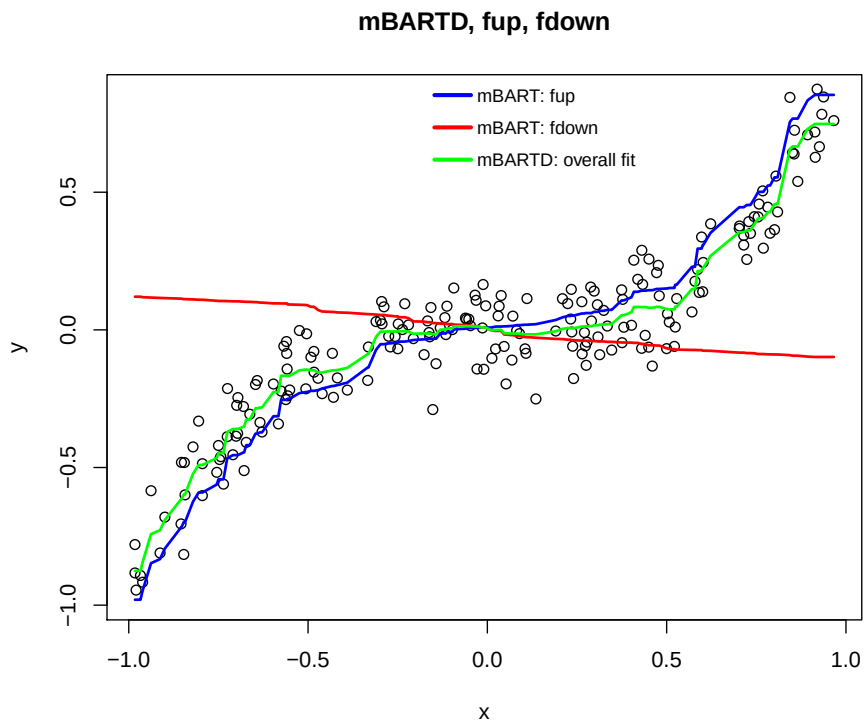
- constrained to be monotone up in the x_1 direction, and
- constrained to be monotone down in the x_2 direction.

We are effectively estimating monotone projections of $f^*(x_1, x_2)$ onto the x_1 and x_2 axes

- $P_{[x_1]} f^*(x_1, x_2) = f_{up}(x_1)$
- $P_{[x_2]} f^*(x_1, x_2) = f_{down}(x_2)$

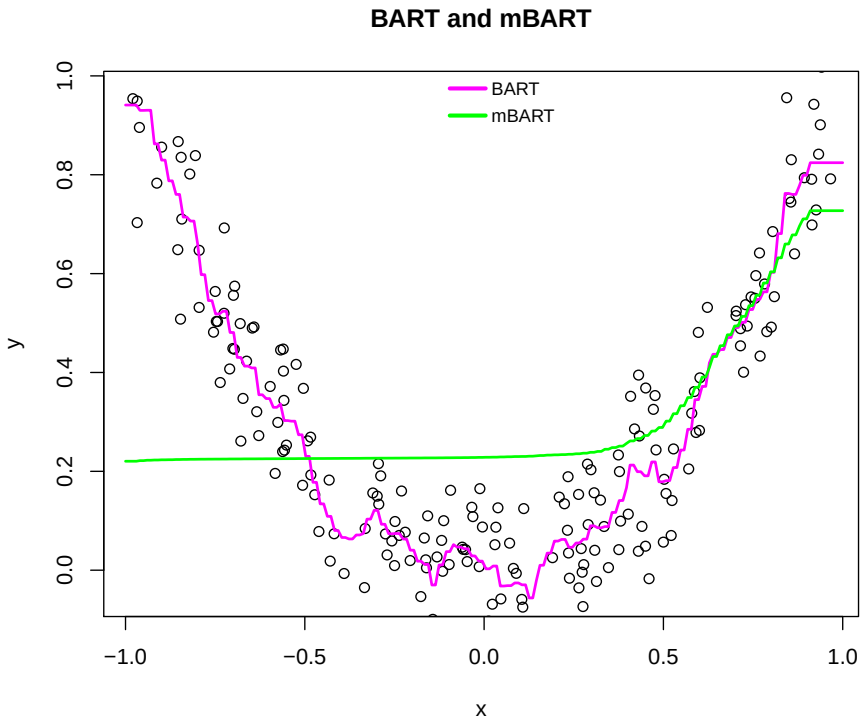
Example: Suppose $Y = x^3 + \epsilon$.

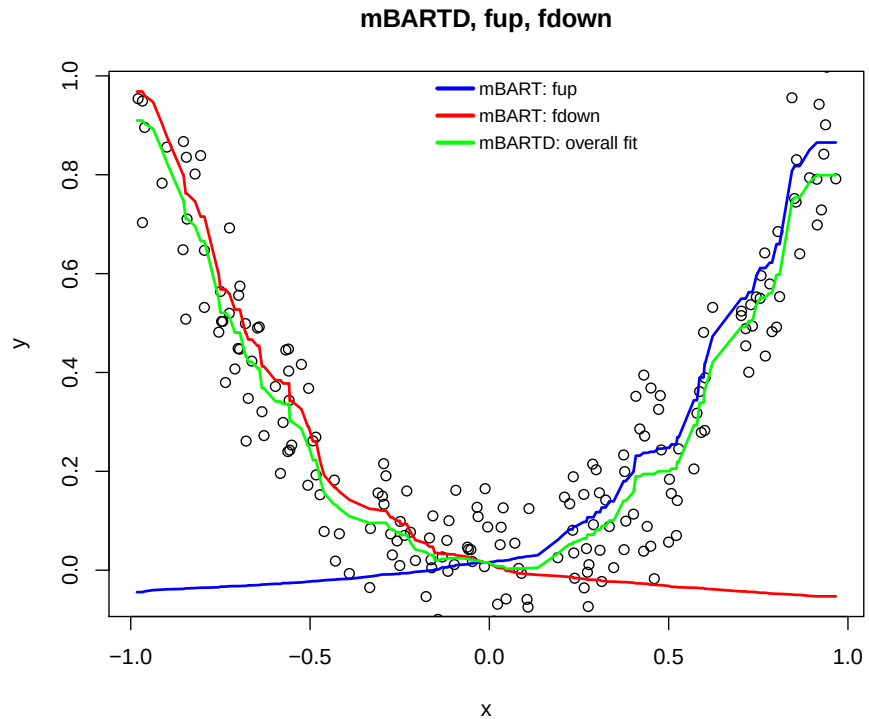




Note that $\hat{f}_{down} \approx 0$ (the red in the right plot), as we would expect when f is monotone up.

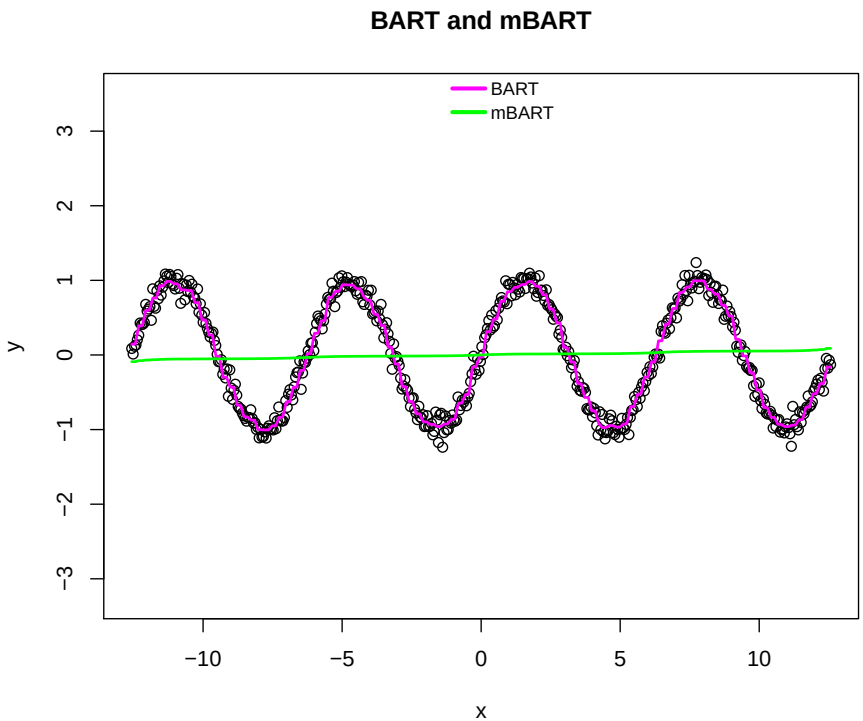
Example: Suppose $Y = x^2 + \epsilon$.

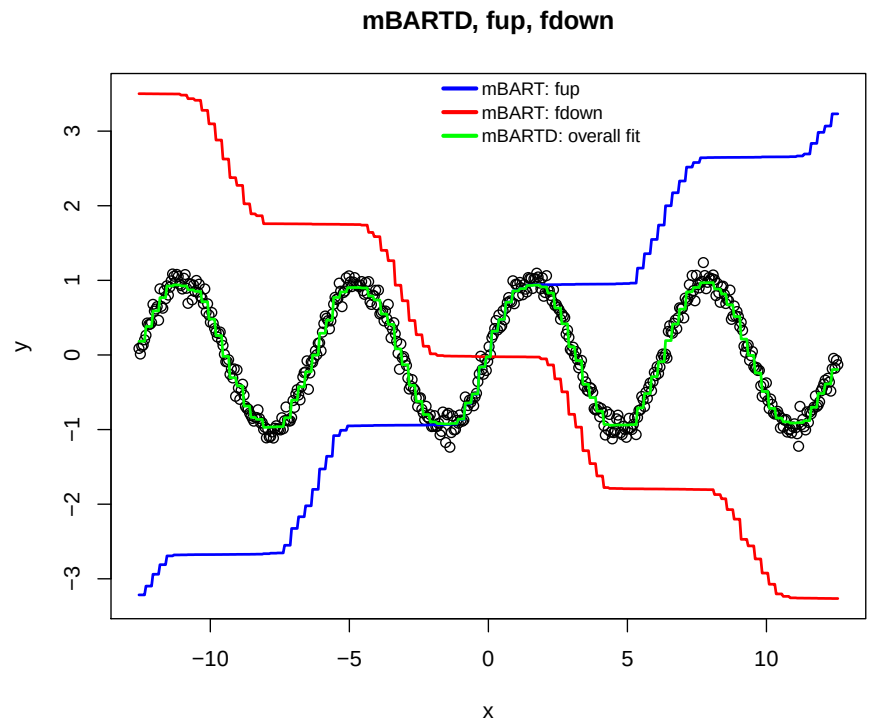




- On the left, BART is good, but simple mBART is not.
- On the right, $\hat{f}_{up}$ and $\hat{f}_{down}$ are spot on.
- And mBARTD $= \hat{f}_{up} + \hat{f}_{down}$ seems better than BART!

Example: Suppose $Y = sin(x) + \epsilon$.





- BART is great, but simple mBART reveals nothing.
- $\hat{f}_{up}$ and $\hat{f}_{down}$ have discovered the monotone decomposition.
- And mBARTD $= \hat{f}_{up} + \hat{f}_{down}$ is great too.

To extend this approach to multidimensional x, we simply duplicate each and every component of x !!!

Example: House Price Data

Let's look at a simple real example where y = house price, and x = three characteristics of each house.

```
> head(x)
     nbhd size brick
[1,]    2 1.79     0
[2,]    2 2.03     0
[3,]    2 1.74     0
[4,]    2 1.98     0
[5,]    2 2.13     0
[6,]    1 1.78     0
> dim(x)
[1] 128   3
> summary(x)
      nbhd            size           brick
 Min.   :1.000   Min.   :1.450   Min.   :0.0000
 1st Qu.:1.000   1st Qu.:1.880   1st Qu.:0.0000
 Median :2.000   Median :2.000   Median :0.0000
 Mean   :1.961   Mean   :2.001   Mean   :0.3281
 3rd Qu.:3.000   3rd Qu.:2.140   3rd Qu.:1.0000
 Max.   :3.000   Max.   :2.590   Max.   :1.0000
> summary(y)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   69.1   111.3   126.0   130.4   148.2   211.2
```

y: dollars (thousands).

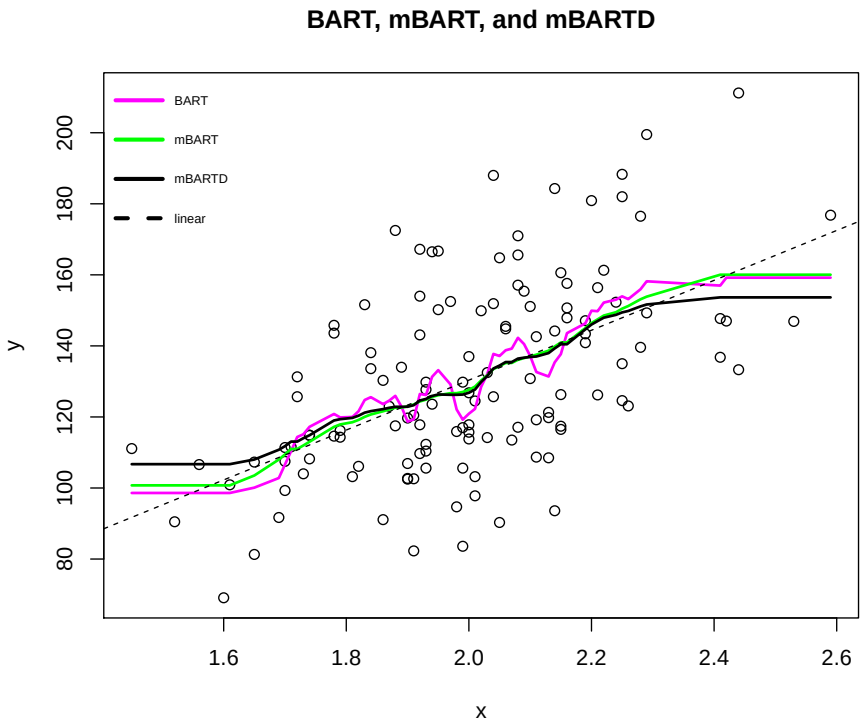
x: nbhd (categorial), size (sq ft thousands), brick (indicator).

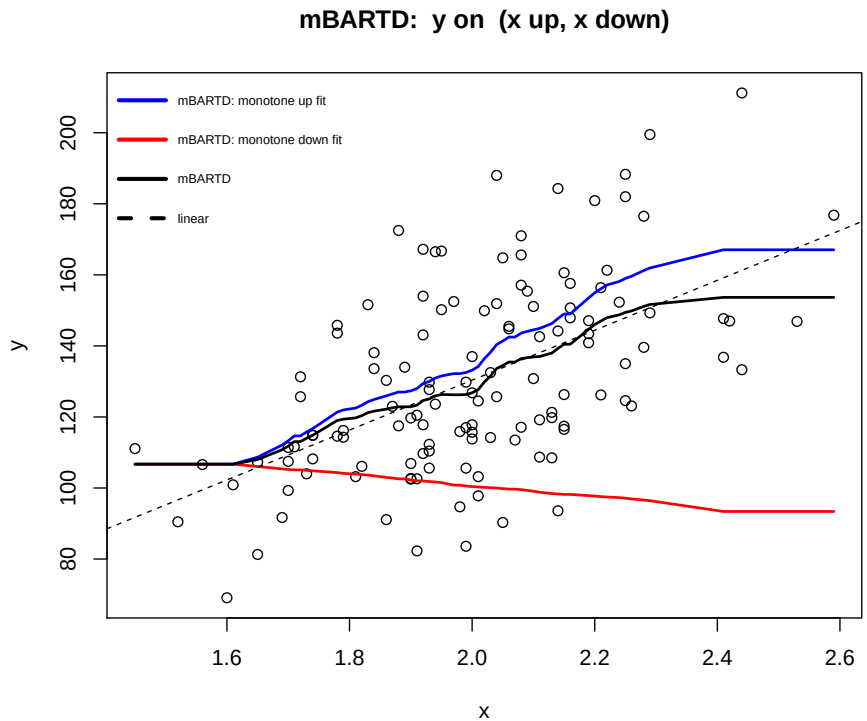
```
Call:
lm(formula = price ~ nbhd + size + brick, data = hdat)

Residuals:
    Min      1Q  Median      3Q     Max
-30.049  -8.519   0.137   7.640  36.912

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   18.725     10.766   1.739   0.0845 .
nbhd2          5.556      2.779   1.999   0.0478 *
nbhd3         36.770      2.958  12.430  < 2e-16 ***
size          46.109      5.527   8.342 1.25e-13 ***
brickYes      19.152      2.438   7.855 1.69e-12 ***
---

Residual standard error: 12.5 on 123 degrees of freedom
Multiple R-squared:  0.7903,Adjusted R-squared:  0.7834
F-statistic: 115.9 on 4 and 123 DF,  p-value: < 2.2e-16
```

If the linear model is correct, we are monotone up in all three variables.

Remark: For the linear model we have to dummy up *nbhd*, but for BART and mBART we can simply leave it as an ordered numerical categorical variable.

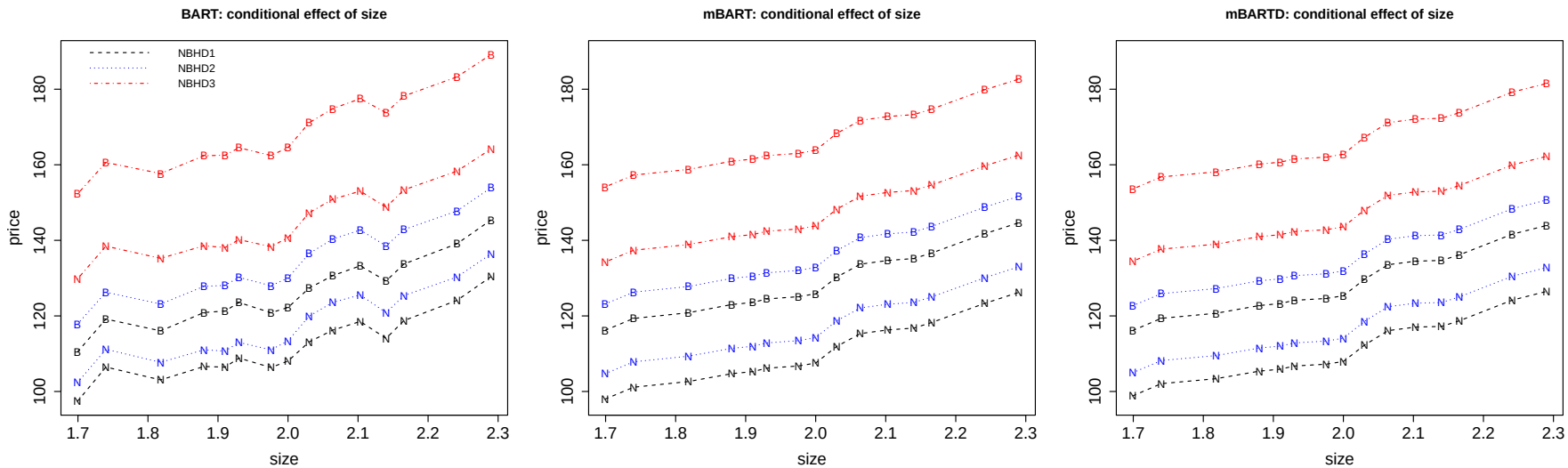
Just using $x = size$, $y = price$ appears to be marginally increasing in *size*. ($\hat{f}_{down} \approx 0$).





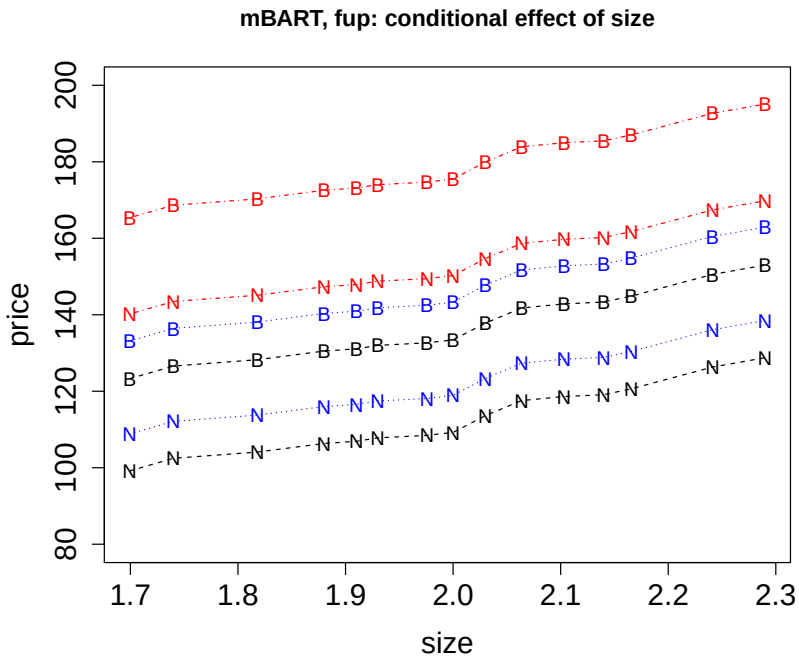
mBART and mBARTD seem much better than BART.

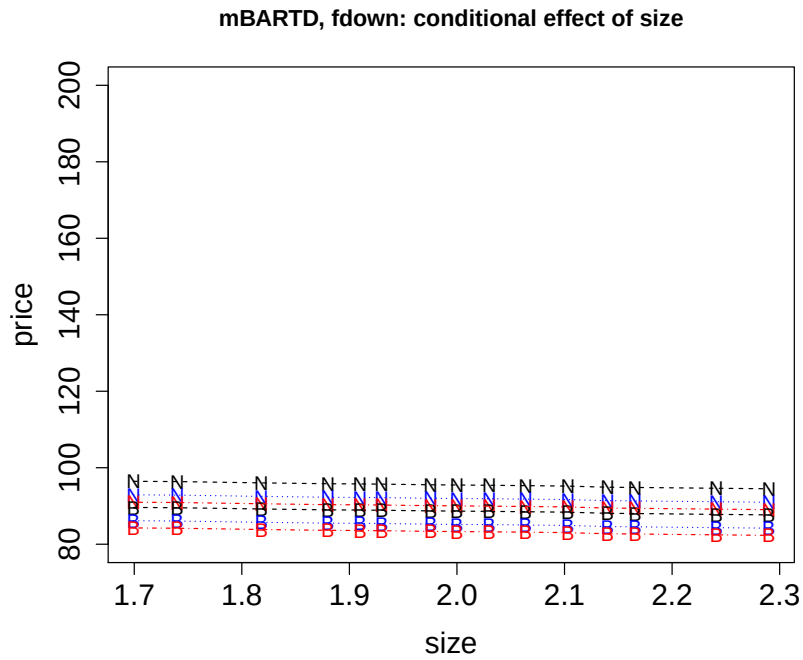
Let's now look at the effect of *size* conditionally on the six possible values of (*nbdh*, *brick*)



The conditionally monotone effect of *size* is becoming clearer!

And finally, the effect of *size* conditionally on the six possible values of (*nbdh*, *brick*) via $\hat{f}_{up}$ and $\hat{f}_{down}$





Price is clearly conditionally monotone up in all three variables!

By simultaneously estimating $\hat{f}_{up} + \hat{f}_{down}$, we have discovered monotonicity without any imposed assumptions!!!

Concluding Remarks

- mBARTD $= \hat{f}_{up} + \hat{f}_{down}$ provides an *assumption free approach* for the discovery of the monotone components of f in multidimensional settings.
- Discovering such regions of monotonicity may of scientific interest in real applications.
- We have used informal variable selection to identify the monotone components here. More formal variable selection can be used in higher dimensional settings.
- As a doubly adaptive shape-constrained regularization approach,
  - mBARTD will adapt to mBART when monotonicity is present,
  - mBARTD will adapt to BART when monotonicity absent,
  - mBARTD seems at least as good and maybe better, than the best of mBART and BART in general.

Concluding Remarks

- Having come to be widely regarded as a successful approach for Bayesian machine learning, new extensions and generalizations of BART are flourishing.

- For a wonderful recent survey of many of these developments, see Hill, Linero and Murray (2020) in the *Annual Review of Statistics and its Applications*.

Happy Birthday Ed!

A true leader!