Monotonically Constrained Bayesian Additive Regression Trees

Robert McCulloch

University of Chicago, Booth School of Business

Joint with: Hugh Chipman (Acadia), Ed George (UPenn, Wharton), Tom Shively (U Texas, McCombs)

SBIES

May 3, 2013

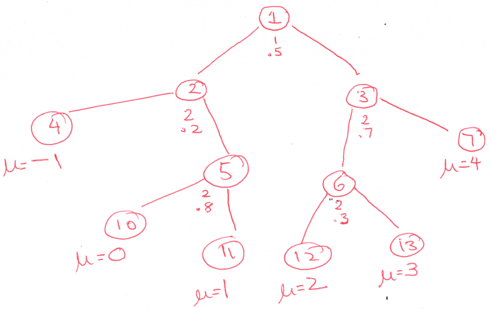
What is a Single Tree Model?

$x = (x_1, x_2)$.

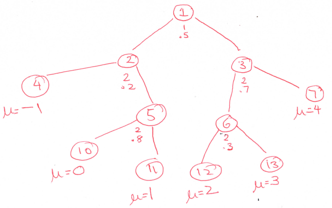
Drop an x down the tree, when it hits bottom, a mean level μ is waiting for it.

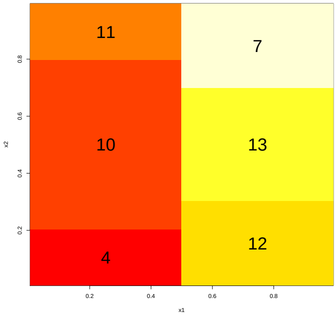
Numbers in circles are node ids.

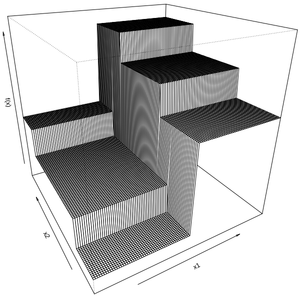
Below node is a decision rule, e.g. 1,.5 means go left if $x_1 < .5$ and right otherwise.



Below each bottom node is the mean level μ for x arriving at that bottom node.

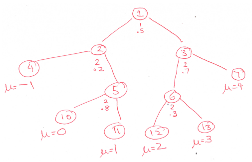


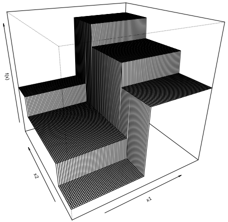




Three different views of a bivariate single tree.

Given $x = (x_1, x_2, \ldots, x_k)$, we can drop x down the tree and get a number.





We denote this function by

$$g(x; T, M)$$

T: the tree structure (including the decision rules)
M: $(\mu_1, \mu_2, \ldots, \mu_b)$, the μ values at the b bottom nodes.

Our single tree model is then

$$Y = g(x; T, M) + \epsilon$$

What is BART?

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

$$Y = g(x; T_1, M_1) + g(x; T_2, M_2) + \ldots + g(x; T_m, M_m) + \epsilon$$

Each (T_i, M_i) denotes a single tree.

$m = 200, 1000, \ldots, big, \ldots.$

T is the sum of all the corresponding μ's at each bottom node from each of m trees plus error.

Such a model combines additive and interaction effects.

Complete the Model with a Regularization Prior

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

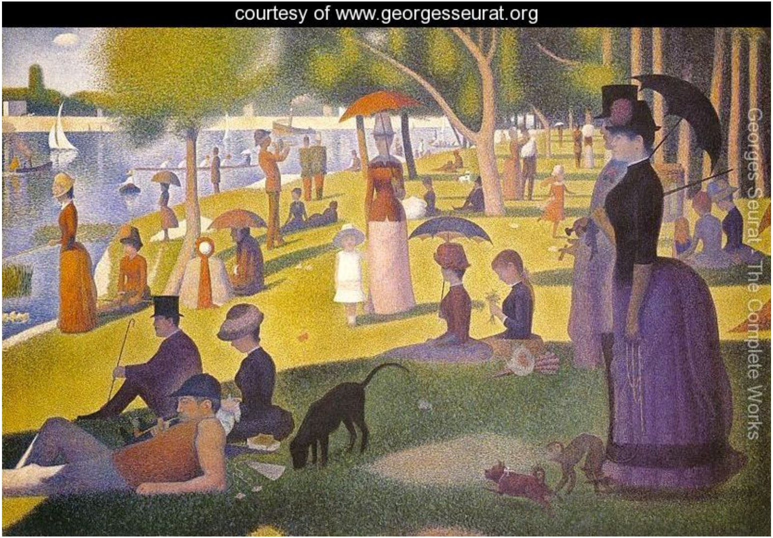
π wants:

- Each T small.
- Each μ small.
- "nice" σ (smaller than least squares estimate).

We refer to π as a regularization prior because it keeps the overall fit from getting "too good".

In addition, it keeps the contribution of each $g(x; T_i, M_i)$ model component small, each component is a "weak learner".

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



Nice things about BART:

- don't have to think about x's (compare: add x_j^2 and use lasso).
- don't have to prespecify level of interaction (compare: boosting in R)
- competitive out-of-sample.
- stable MCMC.
- stochastic search.
- "simple" prior.
- uncertainty.
- big p and/or big n.

Monotonically Constrained BART

We attack the basic problem of estimating a multivariate function constrained to be monotonic.

In a nutshell we:

- use BART, function is a sum of single trees.
- define what it means for each tree to be monotonically constrained
- hence the sum is constrained.
- devise an MCMC algorithm in the constrained space.

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.

But,

we still use the BART/boosting approach to modeling with trees:

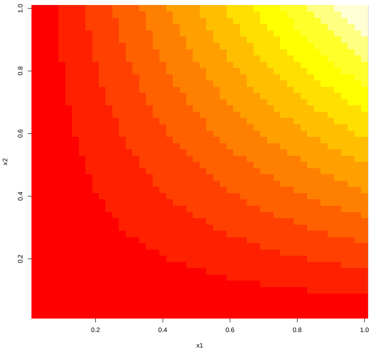
complex montonic functions are built as the sum of many single tree models, each of which is monotonic.

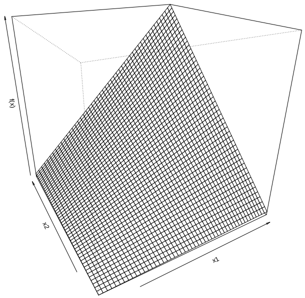
Example: Product of two x's

Let's try a very simple simulated example:

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

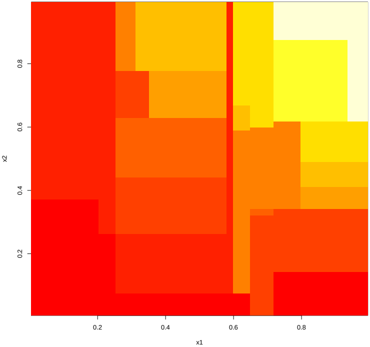
Here is the plot of the true function $f(x_1, x_2) = x_1\, x_2$

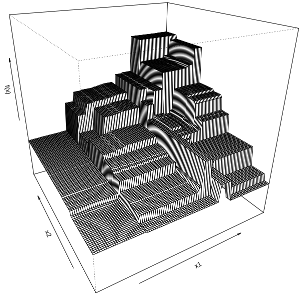




First we try a single (just one tree), unconstrained tree model.

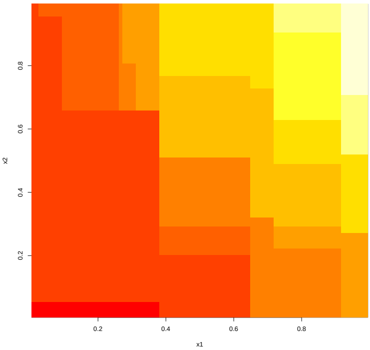
Here is the graph of the fit.

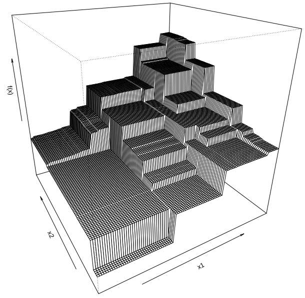




The fit is not terrible, but there are some aspects of the fit which violate monotonicity.

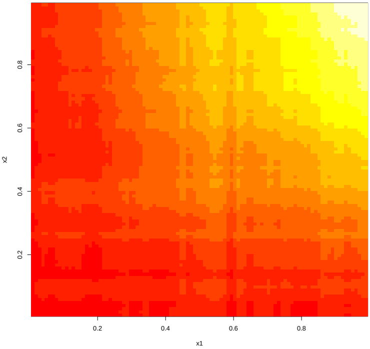
Here is the graph of the fit with the monotone constraint:

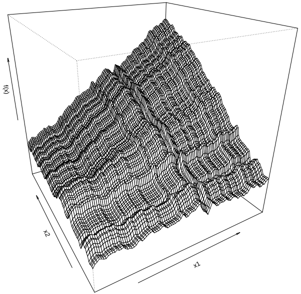




We see that our fit is monotonic, and more representative of the true f.

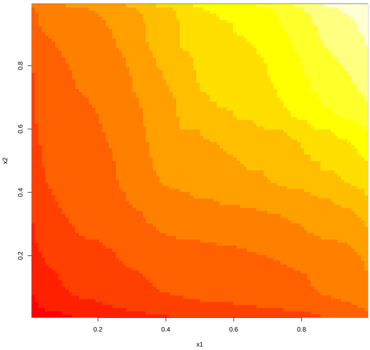
Here is the unconstrained BART fit:

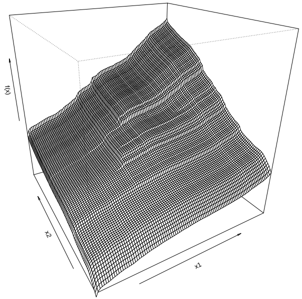




Much better (of course) but not monotone!

And, finally, the constrained BART fit:





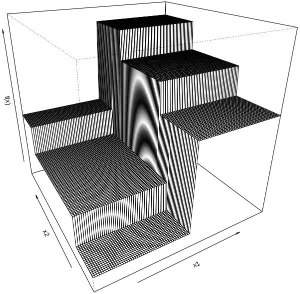
NB!

Same method works with any number of x's!

Monotonic Trees

How do we make a single tree monotonic?

We say this function



is monotonic because,

$$g(x_1, x_2, \ldots, x_i + \delta, x_{i+1}, \ldots, x_k; T, M)$$
$$\geq g(x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_k; T, M), \quad \delta > 0.$$

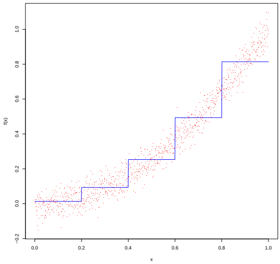
We take the condition

$$g(x_1, x_2, \ldots, x_i + \delta, x_{i+1}, \ldots, x_k; T, M) \\ \geq g(x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_k; T, M), \quad \delta > 0.$$

as our definition.

How do we express this condition in a language trees can understand?

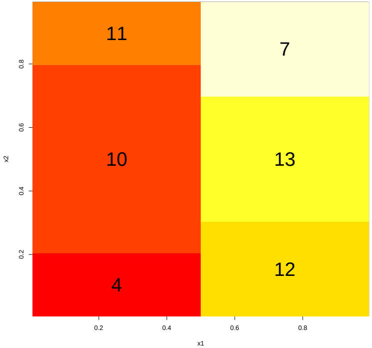
With just one x variable, we can easily see what to do:



- each flat section of f corresponds to a bottom node and a region in x space. With one x, these disjoint regions are intervals.
- for any bottom node, there may be a neighboring region above and and a neighboring region below.
- the mean level for the any bottom node must be greater than that of a below neighbor, and less than that of an above neighbor.

We will:

- Say what we mean for a bottom node to be a below(above) neighbor of a given bottom node.
- Constrain the mean level of a node to be greater than those of it below neighbors and less than those of its above neighbors.



- node 7 is disjoint from node 4.
- node 10 is a below neighbor of node 13.
- node 7 is an above neighbor of node 13.

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

You can code this idea up for general trees!

Note:

For any bottom node, we can figure out the constraint interval for the mean level μ of that bottom node given the rest of the tree.

Above your belows, below your aboves.

Because we will be doing an MCMC and only making local changes, this will be enough.

That is, we don't have to "understand" the constrained set of

$$(\mu_1, \mu_2, \ldots, \mu_B),$$

for a tree with B bottom nodes, and μ_i the mean level of bottom node i.

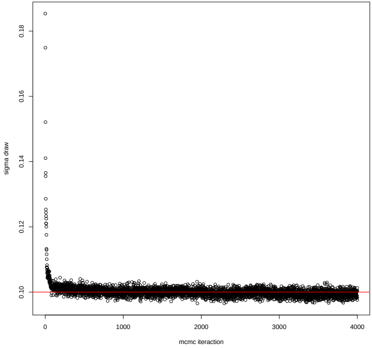
A 5-Dimensional Example

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

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

We simulated 5,000 observations, with $\sigma = .1$.

Here are the MCMC draws of sigma:



The horizontal (red) line is drawn at the true value.

We see that the sampler quickly burns in and then varies about the true value.

Now let's look at the fit, both in-sample and out-of-sample.

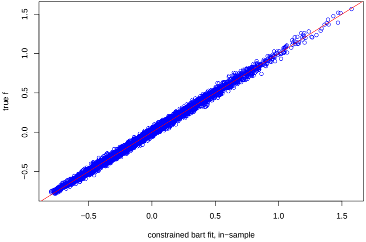
For out-of-sample observations,
we generated two kinds of x's.

- We generated 1,000 x vectors, where each x_i is an independent iid Uniform$(0, 1)$ draw (as for the in-sample training data).
- For each variable, we fixed the other 4 at .5, and then varied the variable across a grid of 20 values from 0 to 1.

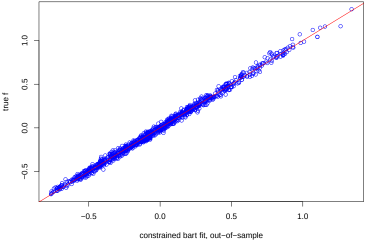
Fits $\hat{f}(x)$ are just the MCMC posterior mean of $f(x)$ for a given x.

"Fit" is given by posterior mean of $f(x)$.

in-sample fit.



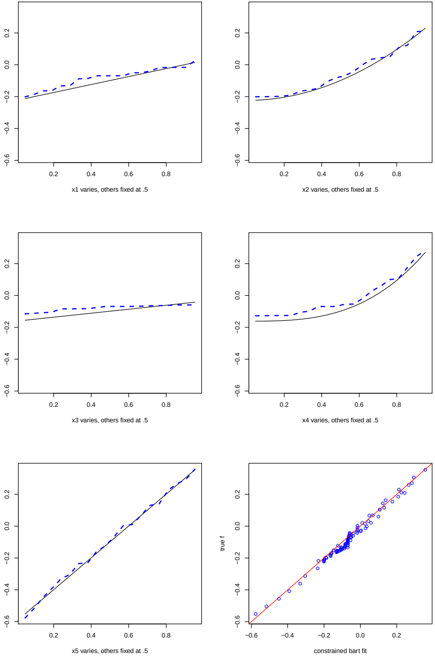
out-of-sample fit.



All but bottom right change one coordinate of x at a time.

Solid black is true, Dashed blue if posterior mean.

Bottom right is $f(x)$ vs $\hat{f}(x)$ (posterior mean) for all out-of-sample change one at a time x.



Example: Real 3-d Example, Orange Juice

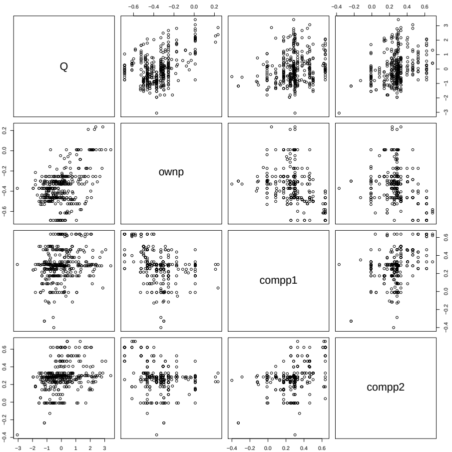
Weekly data on prices and quantity sold of orange juice.

- $Y = Q =$ Quantity sold for the 12oz Minute Maid orange juice.
- $x_1 =$ ownp = - price of 12oz Minute Maid.
- $x_2 =$ compp1 = price of 12oz Florida Gold.
- $x_3 =$ compp2 = price of 12oz Tropicana.

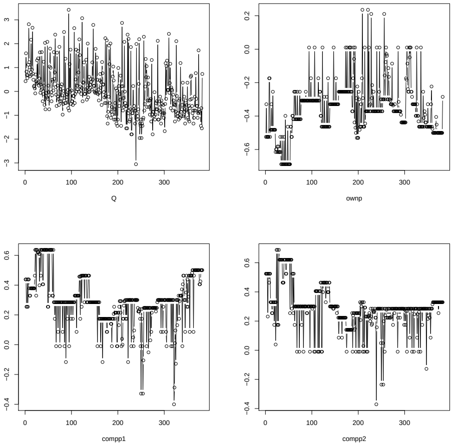
Note:
x_1 is the negative price.

It might make sense to think $E(Y \mid x_1, x_2, x_3)$ is increasing in each x!

All variables are demeaned.



Time series plots of each of the 4 variables:

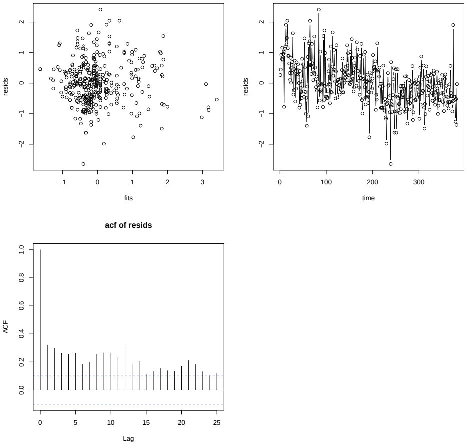


We'll explore regressing Y on the three x's but there may be some specification issues!!

Here is the regression output from Y on all three x's, plus the squares and two-way interactions.

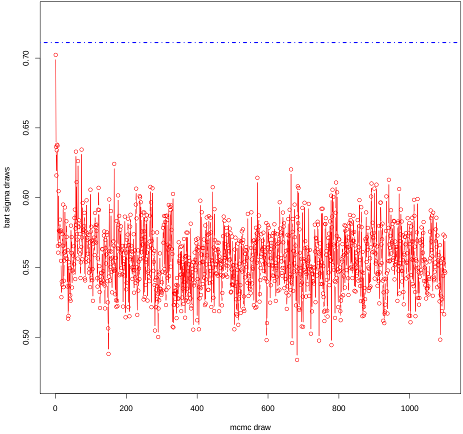
```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.1850     0.2363   0.783 0.434351
ownp          5.1329     0.6706   7.654  1.7e-13 ***
compp1        1.2823     0.7482   1.714 0.087391 .
compp2        3.1318     0.8120   3.857 0.000135 ***
ownpsq        2.9868     1.1322   2.638 0.008688 **
compp1sq      2.3020     1.1264   2.044 0.041690 *
compp2sq      4.0512     1.2346   3.281 0.001131 **
owncomp1      0.1406     2.1408   0.066 0.947688
owncomp2      4.6167     2.2489   2.053 0.040782 *
comp1comp2   -2.6551     1.7691  -1.501 0.134242
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.7111 on 372 degrees of freedom
Multiple R-squared: 0.5362,Adjusted R-squared: 0.525
F-statistic: 47.79 on 9 and 372 DF,  p-value: < 2.2e-16
```

Diagnostic plots for the regression:



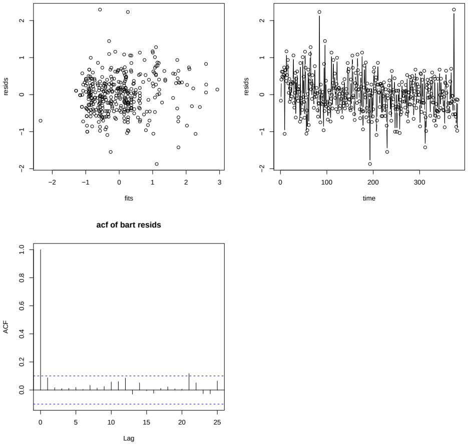
There is time series structure in the problem not being captured by the regression.

red: mcmc draws of σ from BART.
blue line at top: estimate of σ from the regression



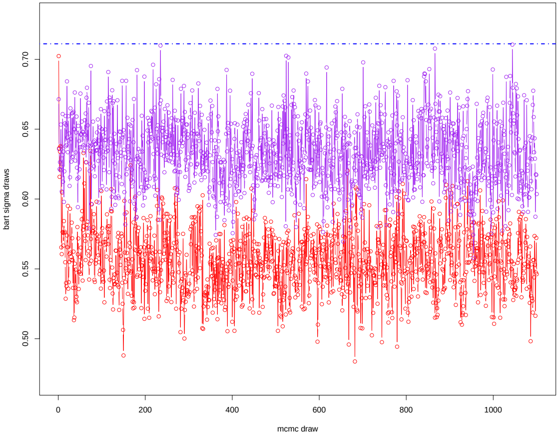
BART claims to have found a much better fit!!

Here are the diagnostic plots for the BART fit:

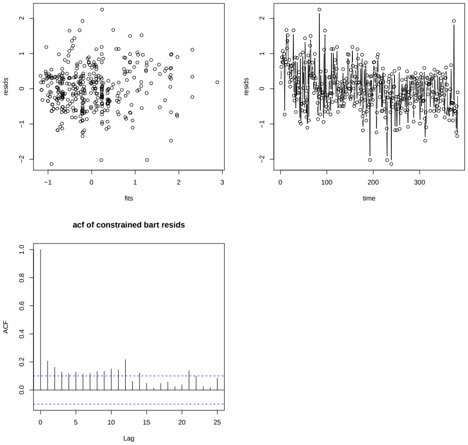


While there appear to be a few outliers, the time series behaviour of the resids is much better!

red: mcmc draws of σ from BART.
blue line at top: estimate of σ from the regression.
purple: mcmc draws of σ from constrained BART.



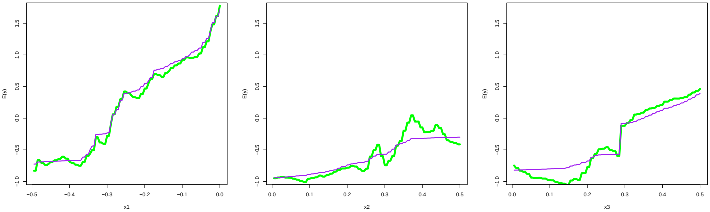
Here are the diagnostic plots for the constrained BART fit.



Still much better than the regression, but not as good as unconstrained.

Here we compare out of sample predictions by fixing two of the three x's at their means and then varying the third on a grid of values.

green: BART
purple: constrained BART



We see that the constraints are indeed enforced: if only one x increases, $E(Y)$ must increase.

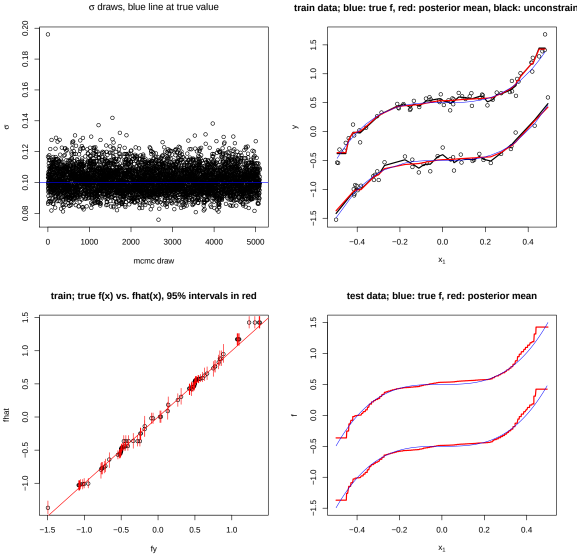
What do we conclude??

While the constrained BART is not as good as the unconstrained, it is a huge improvement of the regression with transformations.

It may well be worth while giving up some in-sample fit to get a model that makes more sense!!

Two x, one binary

$$y = 8x_1^3 + x_2 + \epsilon,\ x_1 \sim U(-.5, .5),\ p(x_2 = -.5) = p(x_2 = .5) = .5.$$



Comments on Algorithm

- BART is based on a sum, and the sum of monotonic is monotonic.
- Can write code to find the constraint interval for the μ of a bottom node given the rest of the tree.
- MCMC works on a single tree at a time.
- MCMC makes local moves so we only have to think about at most two bottom nodes at a time
  $\Rightarrow$
  don't have to understand the full set of constrained $\mu_i, i = 1, 2, \ldots, b$ for b bottom nodes.

BART MCMC

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

plus

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

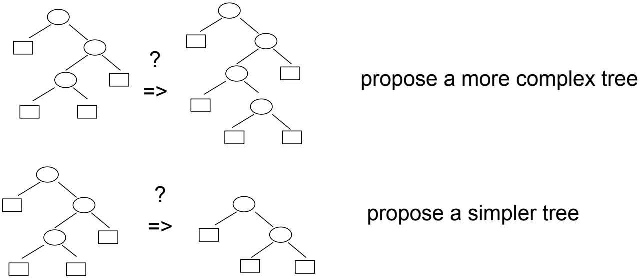
First, it is a "simple" Gibbs sampler:

$$\begin{array}{rcl} (T_i, M_i) & | & (T_1, M_1, \ldots, T_{i-1}, M_{i-1}, T_{i+1}, M_{i+1}, \ldots, T_m, M_m, \sigma) \\ \sigma & | & (T_1, M_1, \ldots, \ldots, T_m, M_m) \end{array}$$

To draw $(T_i, M_i) \,|\, \cdot$ we subract 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 $M \,|\, T$.

To draw T we use a Metropolis-Hastings with Gibbs step.
We use various moves, but the key is a "birth-death" step.



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

Monotone BART Prior and MCMC

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

To impose the constraint we simply condition on the set where each tree gives a montonic function,

$$\pi^c(\theta) \propto \pi(\theta)\, \chi_S(\theta),$$

where $\chi_S(\theta)$ is 1 if *each* tree is montonic.

Note:

We modify the unconstrained prior to prefer bigger trees and then get back to smaller trees after we impose the constraint.

We can't integrate out the μ's, so when we do a birth/death, we have to propose new bottom node μ values as well as the tree modification.

So, for example, in a birth, we have to propose:

- A bottom node to add a rule to.
- A decision rule.
- a μ_L for the new left child and a μ_R for the new right child where (μ_L, μ_R) are such that the new tree gives a monotonic function.