Multidimensional Monotonicity Discovery with mBART

Robert McCulloch
ASU
Rober.McCulloch@asu.edu

Collaborations with:
H. Chipman, **E. George**, T. Shively

Plan

- I) BART (Bayesian Additive Regression Trees)
- II) Monotone BART: mBART
- III) Monotonicity Discovery with mBART

Part I. BART (Bayesian Additive Regression Trees)

The Fundamental Regression Setup:

- Data: n observations of y and $x = (x_1, ..., x_p)$
- Suppose: $Y = f(x) + \epsilon, \quad \epsilon \sim N(0, \sigma^2)$

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 and x for inference.

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

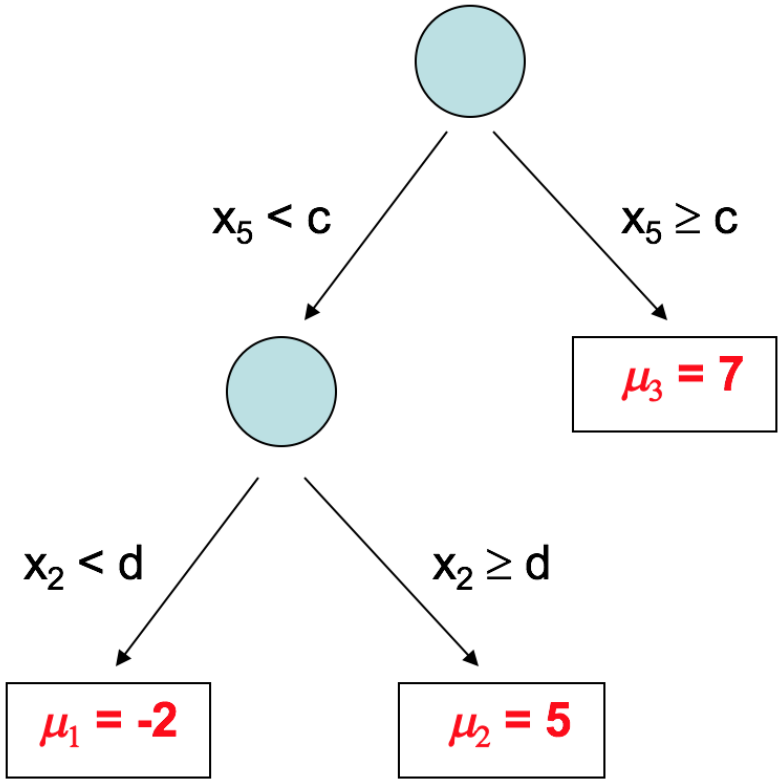
Key calibration: Using y, set $\pi(\theta)$ so that $\text{Var}(f) \approx \text{Var}(y)$.

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:

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

$$\text{“}\theta = (T, M)\text{”}$$

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

Bayesian CART Model Search
Chipman, George, McCulloch (1998 *JASA*), Mallick and Smith (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.

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

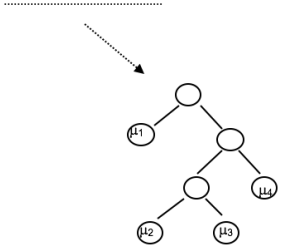
Closed form for $\pi(T \mid y)$ facilitates MCMC stochastic search for promising trees.

Moving on to BART

Bayesian Additive Regression Trees
Chipman, George, McCulloch (2007 *NIPS*, 2010 *AOAS*)

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/Boosting" 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)

- Hyperparameters of π are set so that both the T_j's and the μ_{ij}'s are likely to be small.
- Thus, π keeps the contribution of each $g(x; T_j, M_j)$ small, to explain only a small portion of the fit.

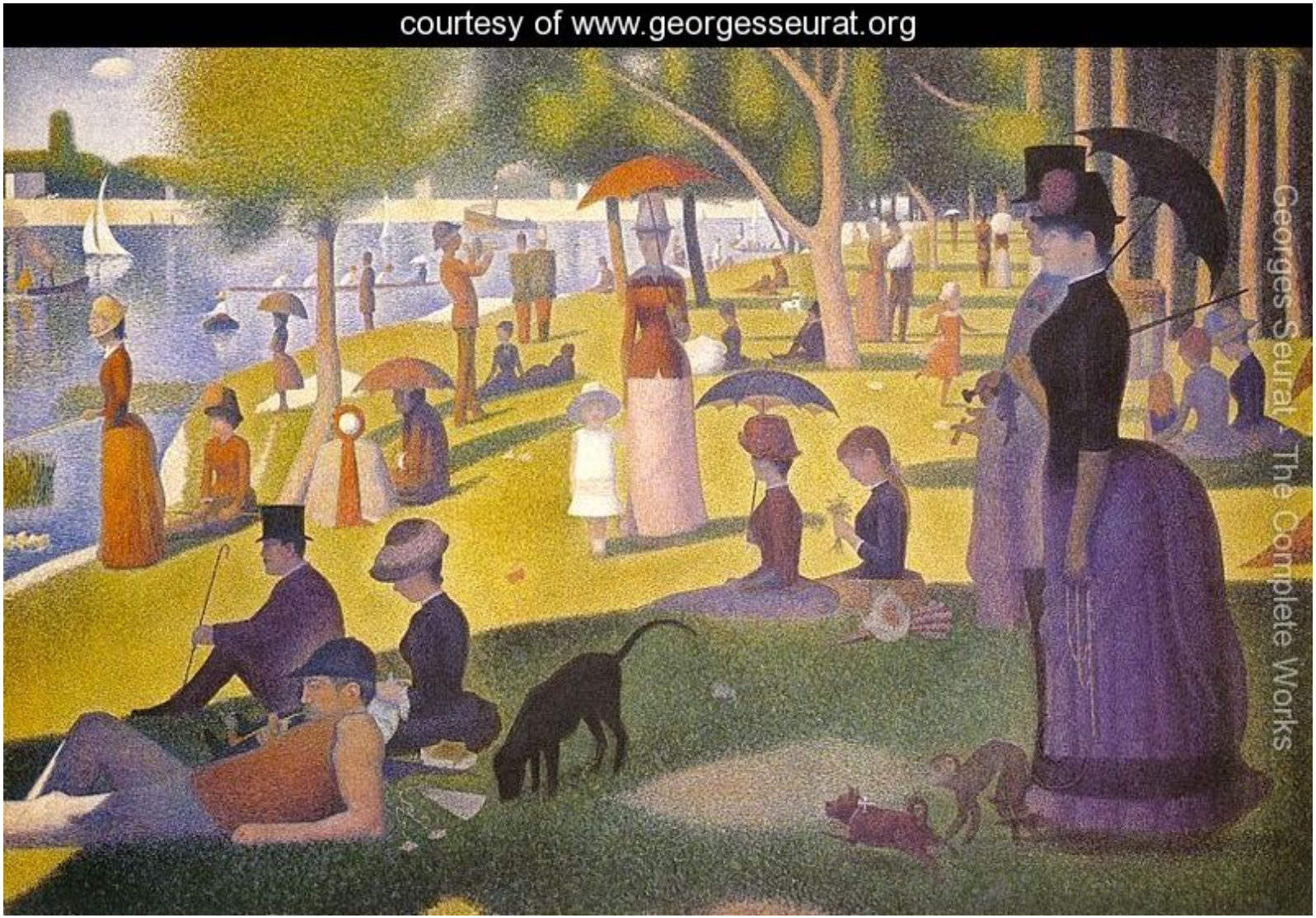
The observed variation of Y is used to guide the hyperparameter settings for the μ and σ priors.

- Note that because μ_{ij}'s iid $\sim N(0, \tau^2)$, we have

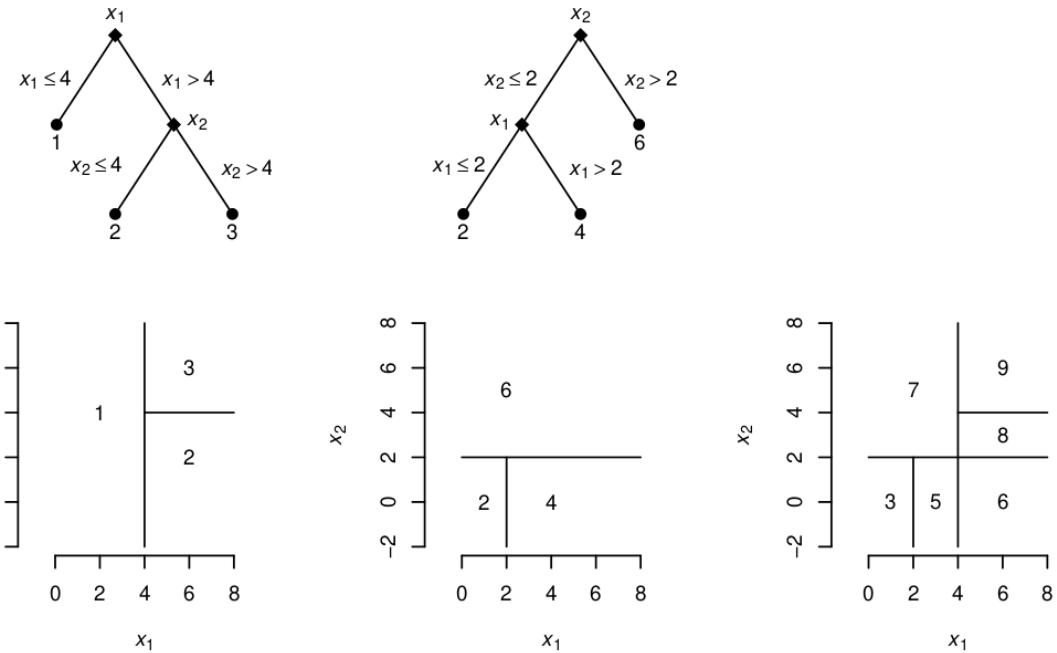
$$Var(Y) \approx Var(\Sigma_{ij}\mu_{ij}) = m\tau^2,$$

which facilitates the calibration of τ^2 for BART's success!

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



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



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

Gradient 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

Some Distinguishing Features of BART

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

BART is NOT Bayesian model averaging of a single tree model

Unlike boosting and random forests, the BART algorithm updates a fixed set of m trees, over and over

Choose m large for best estimation of $E[Y|x]$ and prediction

- More trees yields more approximation flexibility

Choose m small to measure variable importance

- Fewer trees force the x's to compete for entry

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_i to draw T_i and then draw $M_i \mid T_i$.

... 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 yields a new draw from the posterior of f

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

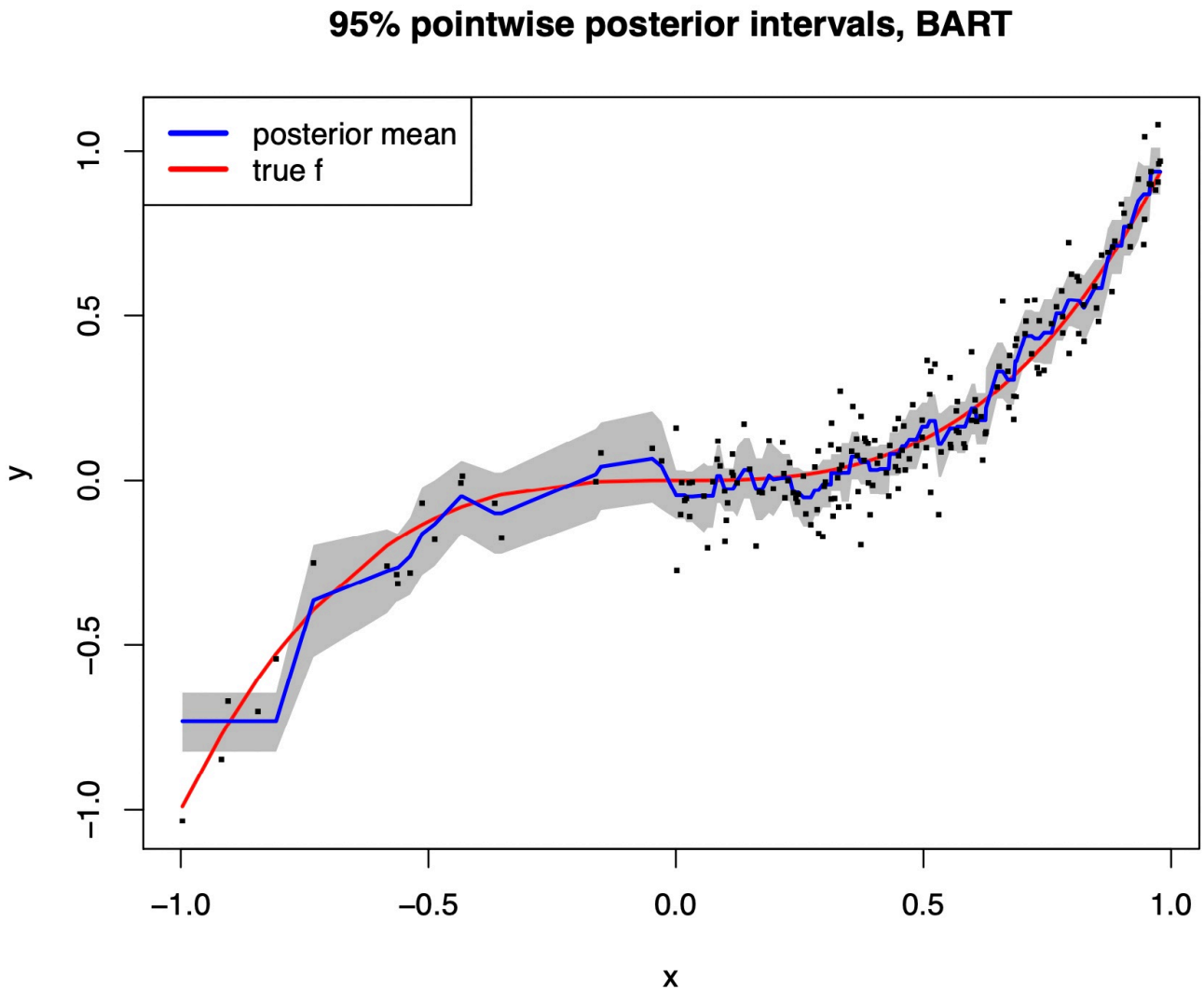
To estimate $f(x)$ we simply average the $\hat{f}_d(\cdot)$ draws at x

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

Can do the same with functionals of f.

Automatic Uncertainty Quantification

A simple simulated 1-dimensional example



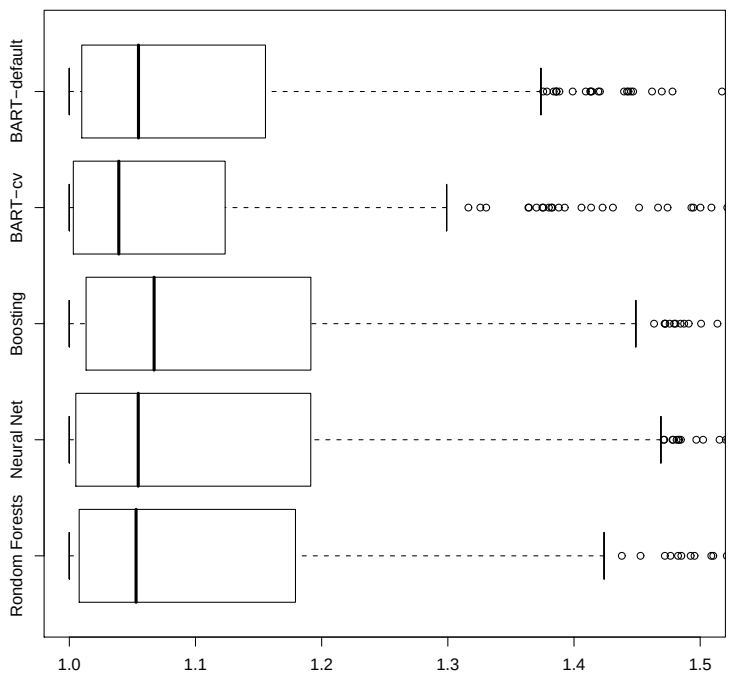
Out of Sample Prediction Comparisons

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 CV on train to pick hyperparameters (except BART-default!).
- Gives 20*42 = 840 **out-of-sample predictions**. For performance comparisons, the RMSE of each method is divided by the smallest RMSE of all.

+ Each boxplot represents 840 predictions for a method
+ 1.2 means you are 20% worse than the best
+ BART-cv best
+ BART-default does amazingly well!!



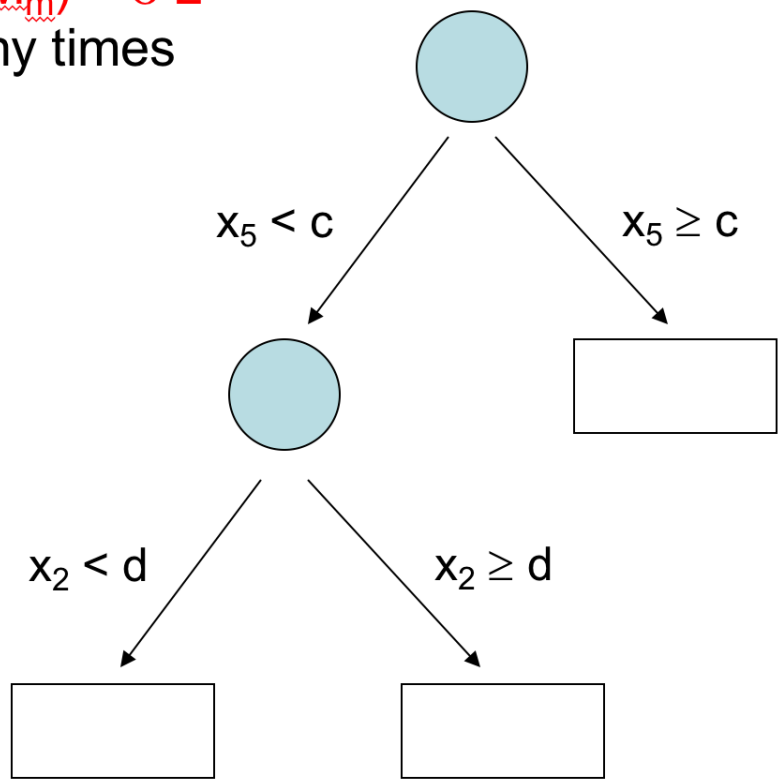
Measuring Variable Importance with BART

When $Y = g(x;T_1,M_1) + \ldots + g(x;T_m,M_m) + \sigma z$ is fit to data, we can count how many times a predictor is used in the trees.

For example, in the tree here, x_2 and x_5 are each used once.

The importance of each x_k can thus be measured by its overall usage frequency.

This approach is most effective when the number of trees m is small.



A Competitive Bottleneck for Entry

BART is an automatic attractor of x's to explain Y

Those x's which most explain Y are attracted most

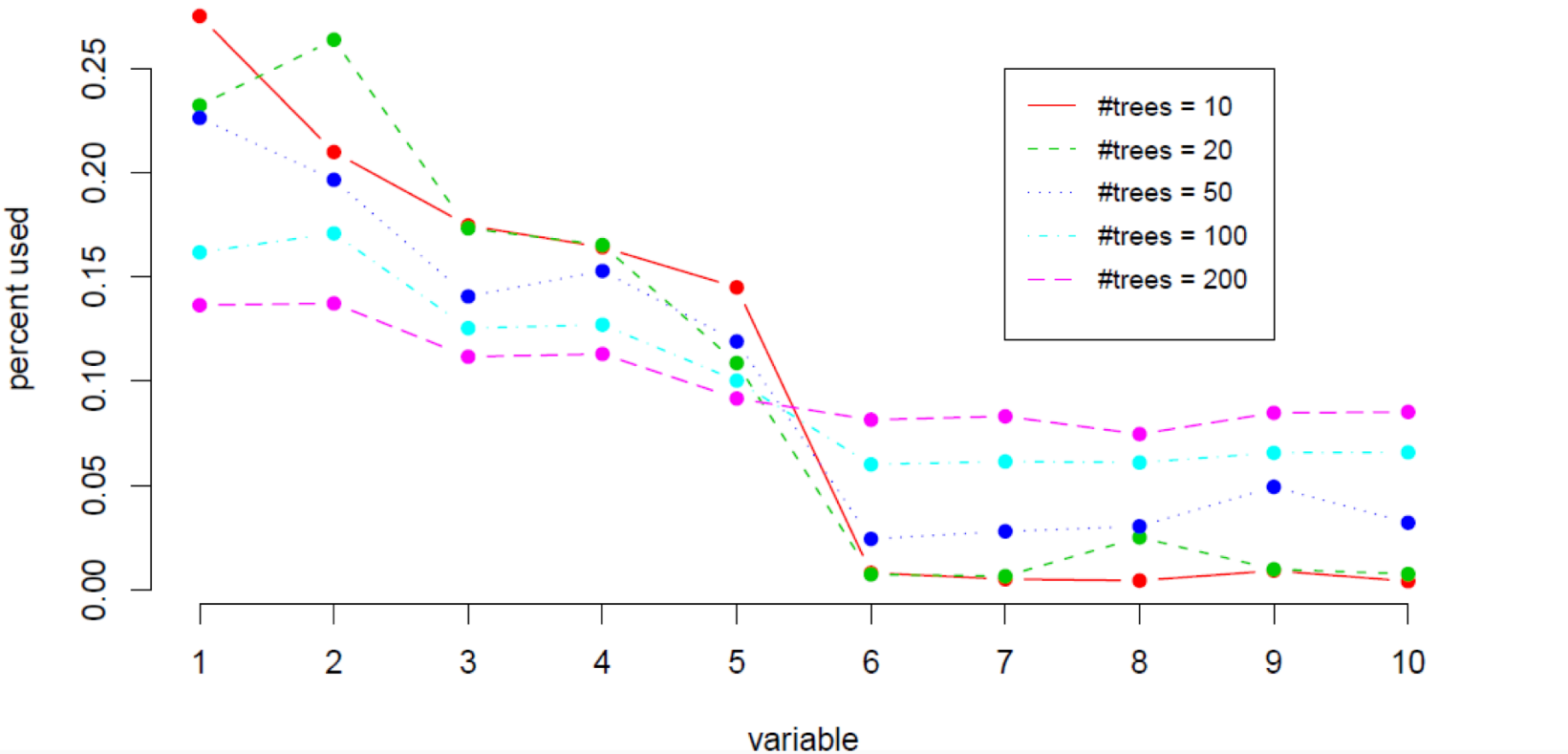
Small number of trees m creates a bottleneck which excludes useless x's

Example: The Friedman Test Function

BART variable importance on data simulated from:

$$Y = 10\sin(\pi x_1 x_2) + 20(x_3 - .5)2 + 10x_4 + 5x_5 + 0x_6 + \cdots + 0x_{10} + \epsilon$$

Variable usage frequencies as the number of trees m is reduced



Part II. Monotone BART - mBART

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

The Key Idea:

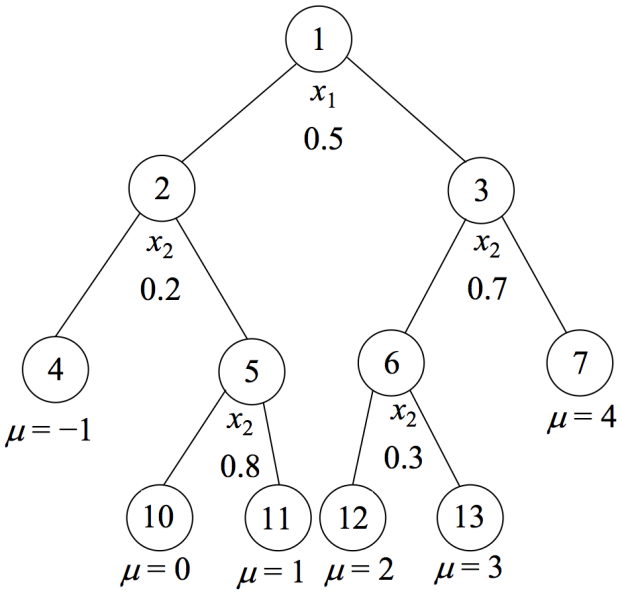
BART approximates a function by a sum of tree functions

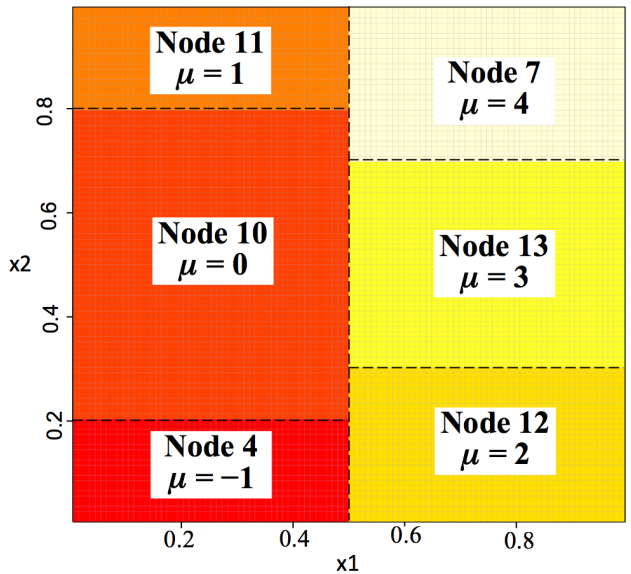
mBART approximates a monotone function by a sum of monotone tree functions

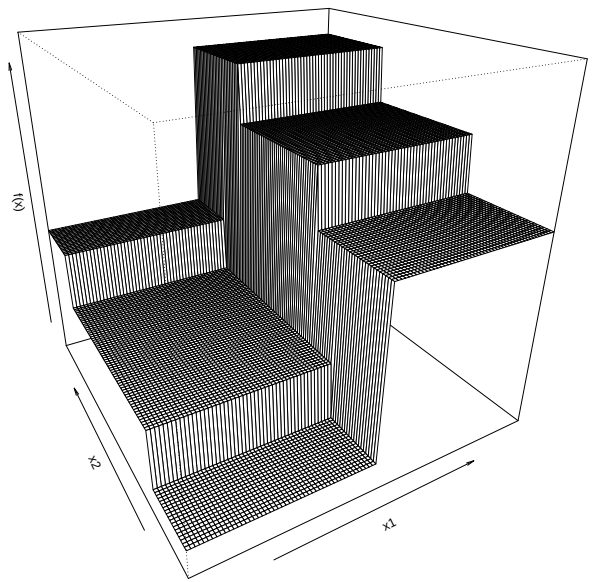
This works because of the obvious fact:

The sum of monotone functions yields a monotone function

An Example of a Monotone Tree Function

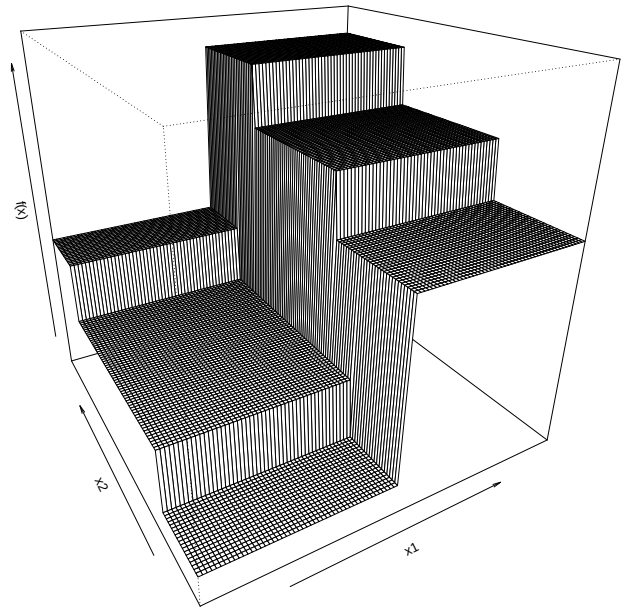






Three different views of a bivariate monotone tree.

In what sense is this tree function monotone?



A tree function $g(x; T, M)$ is said to be *monotone nondecreasing* in x_i if for all x_{-i} and $\delta > 0$,

$$g(x_i,\, x_{-i}\,;\; T, M) \leq g(x_i + \delta,\, x_{-i}\,;\; T, M)$$

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

The mBART Prior

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

Recall the BART parameter

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

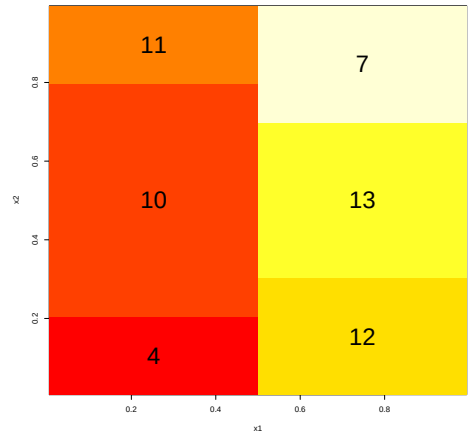
Let $S = \{\theta : \text{each } (T_j, M_j) \text{ is monotone in a desired subset of } x_i's\}$

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 monotone.

Forcing 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".



For example, the mean level of node 13 must be greater than those of 10 and 12 and less than that of node 7.

For any bottom node μ, given the rest of the tree, we can figure out (and easily code) its interval of constraint.

Because we only make local changes via the MCMC algorithm, this criterion suffices for all computations.

The remaining challenge is the construction of a new algorithm which can handle the nonconjugacy of truncated priors on μ's.

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.

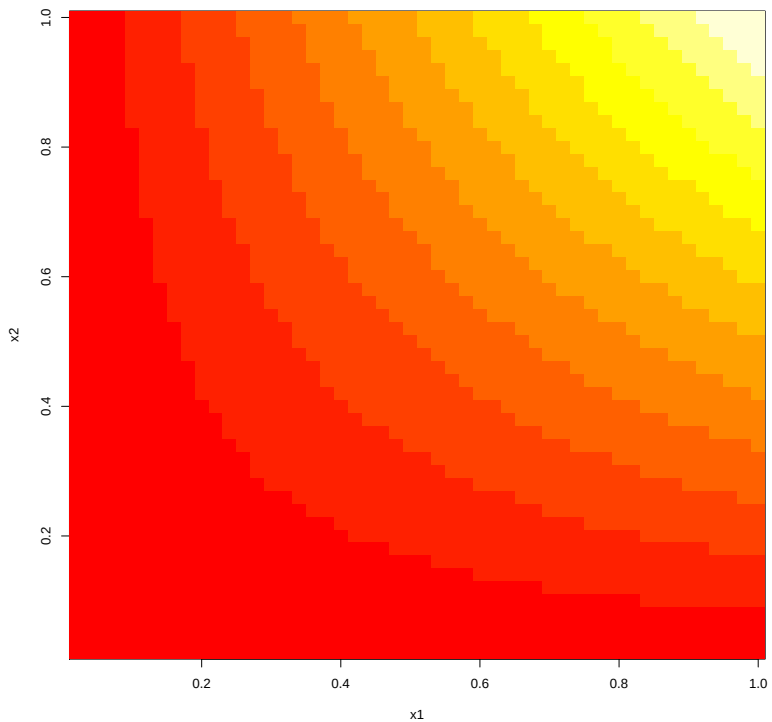
Works for both BART and mBART.

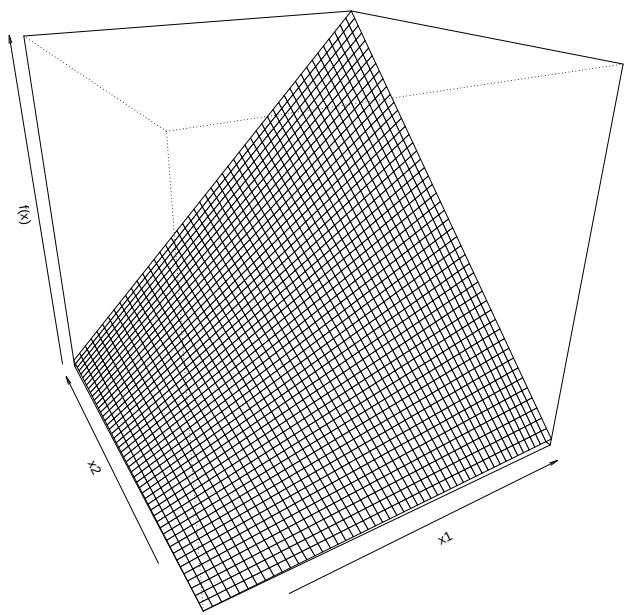
Example: Product of two x's

Let's consider a very simple simulated monotone example:

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

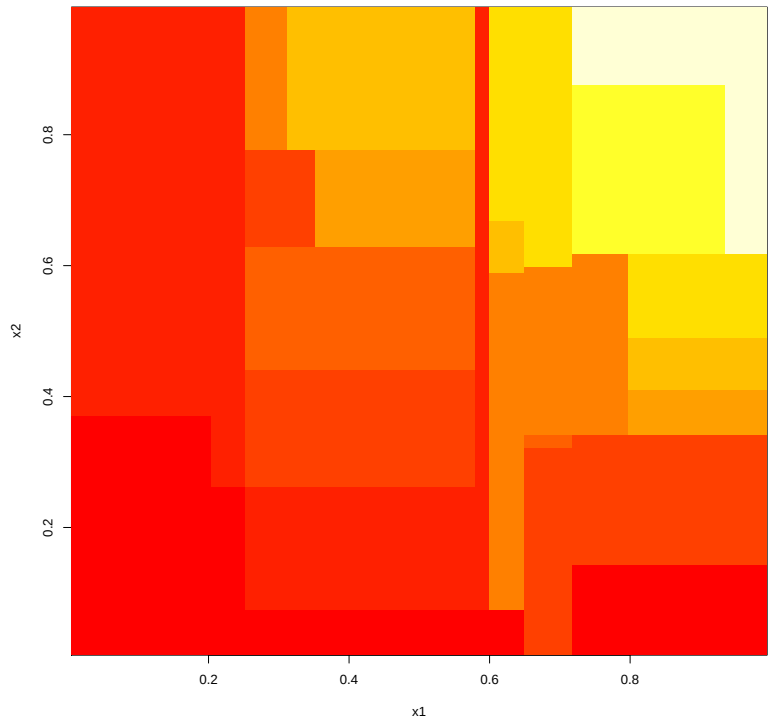
Here is the plot of the true function $f(x_1, x_2) = x_1\, x_2$

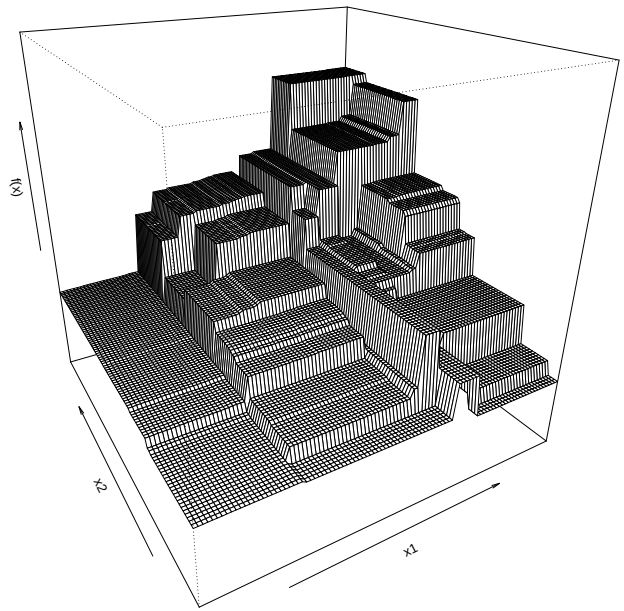




First we try a single (just one tree), unconstrained tree model.

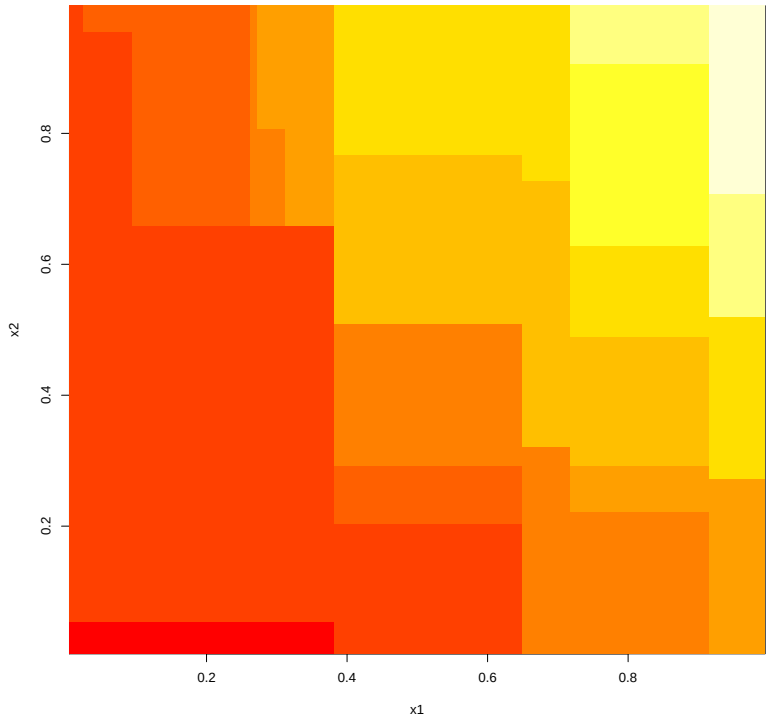
Here is the graph of the fit.

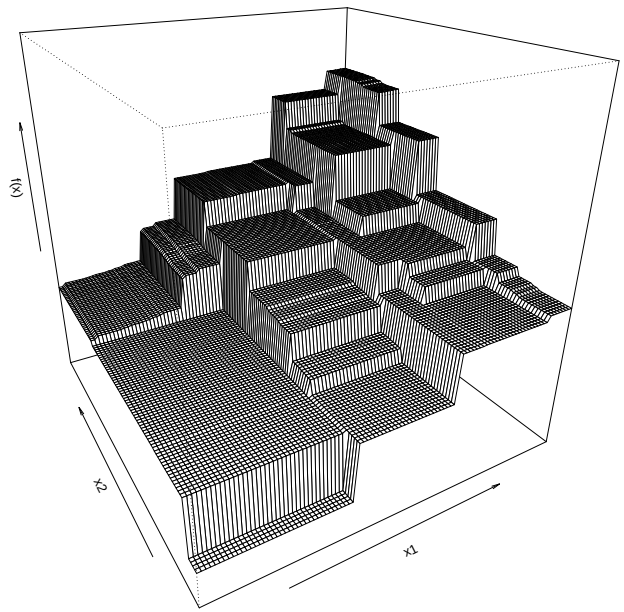




The fit is not terrible, but there are some aspects of the fit which violate monotonicity.

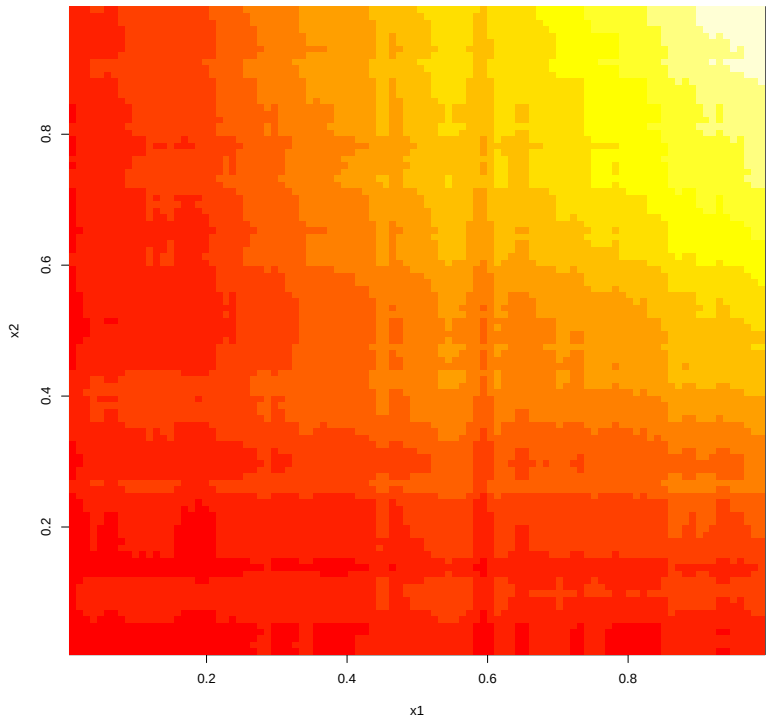
Here is the graph of the fit with the monotone constraint:

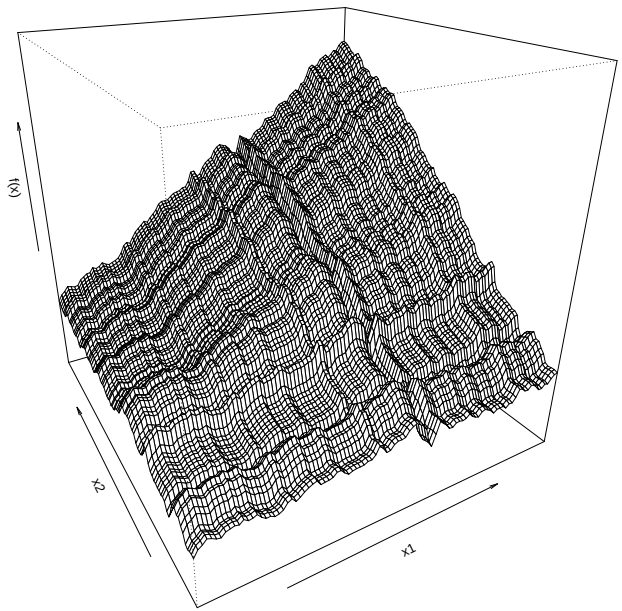




We see that our fit is monotonic, and more representative of the true f.

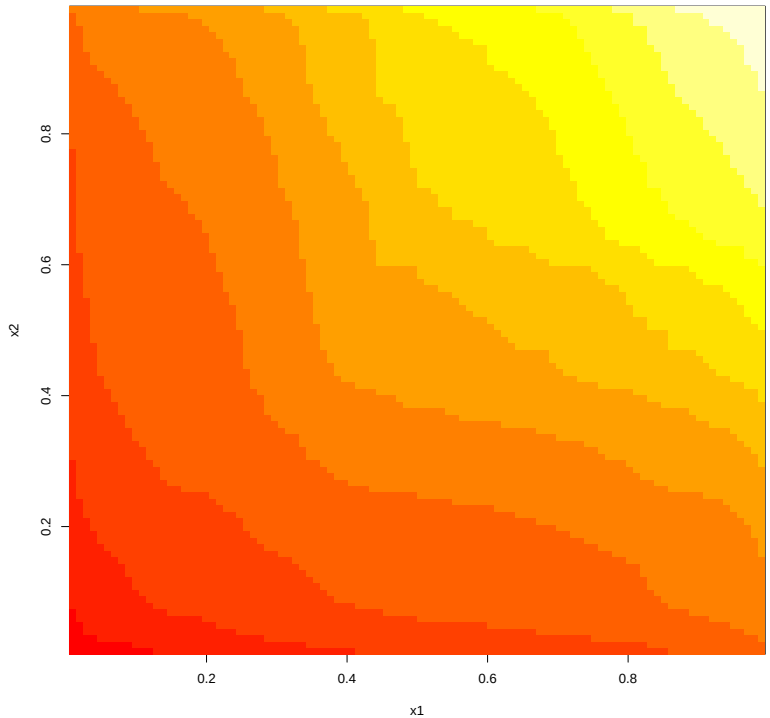
Here is the unconstrained BART fit:

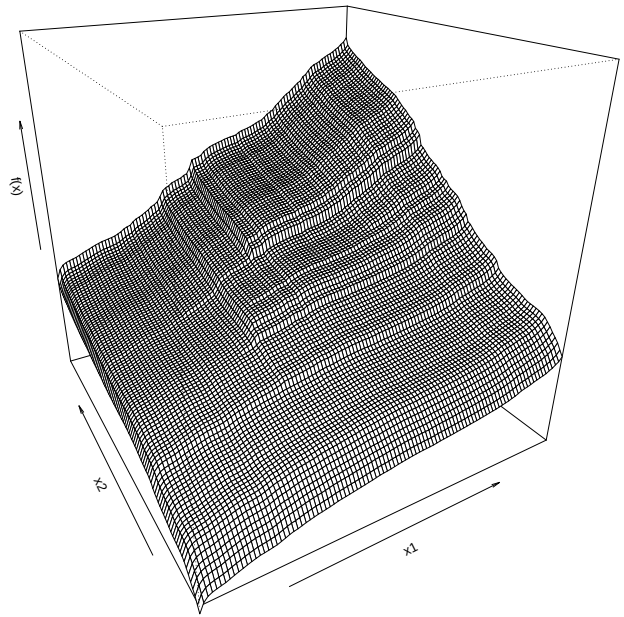




Much better (of course) but not monotone!

And, finally, the constrained BART fit:



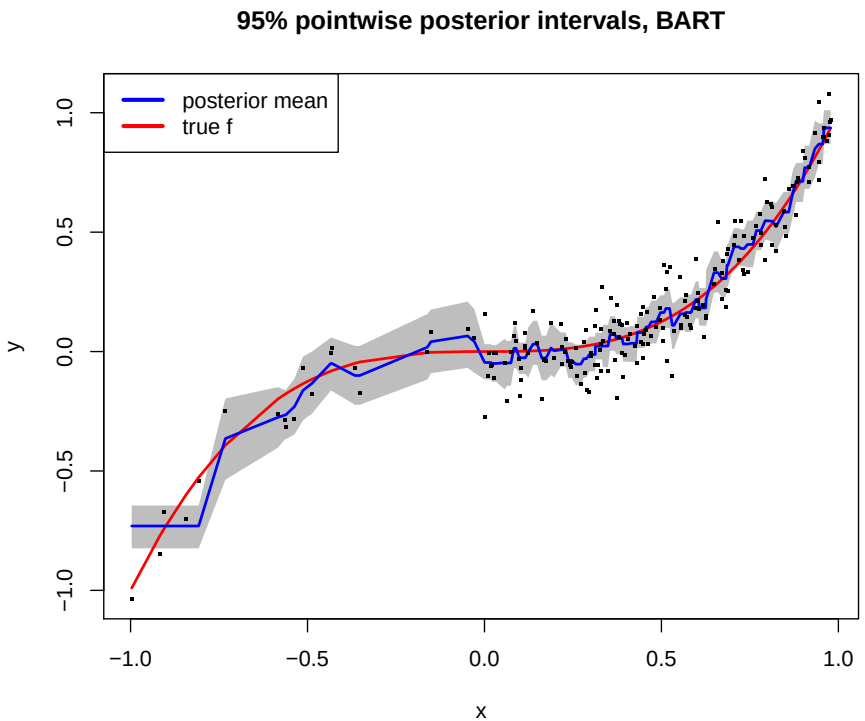


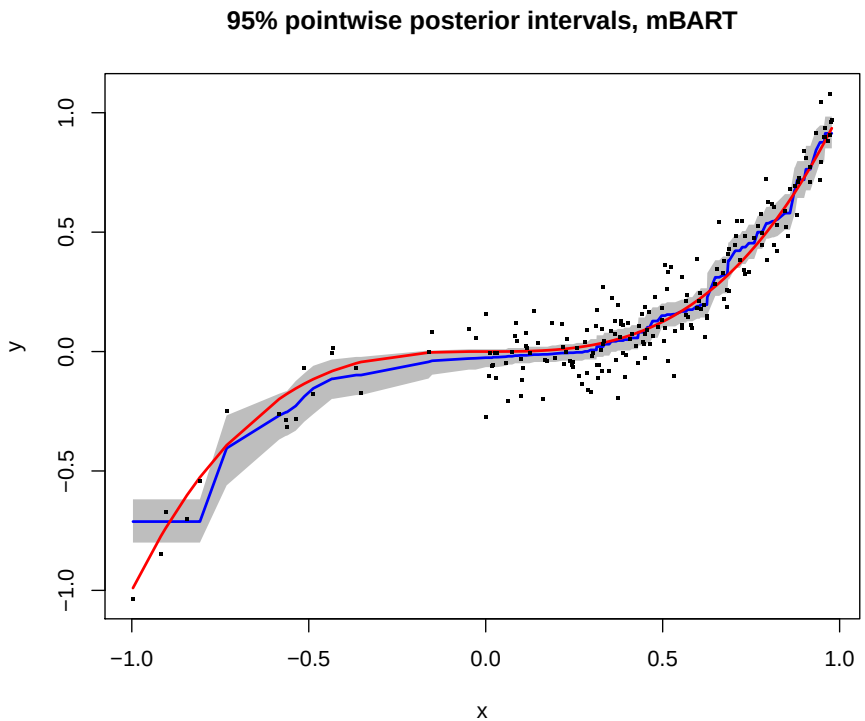
Not Bad!

Same method works with any number of x's!

Automatic Uncertainty Quantification

Revisiting our simple simulated 1-dimensional example





mBART intervals are tighter!

Example: RMSE Reduction by Monotone Regularization

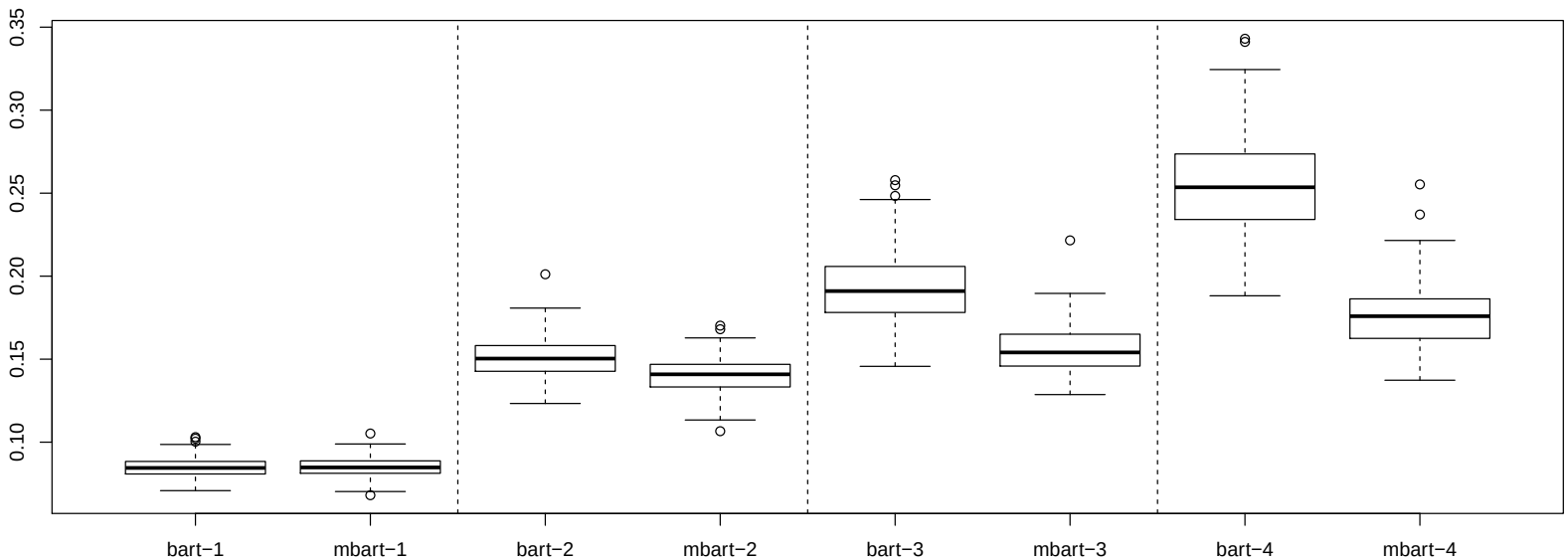
$$Y = x_1\, x_2^2 + x_3\, x_4^3 + x_5 + \epsilon,$$

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

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

RMSE improvement of mBART over 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 to simply compare the fits from BART and mBART.

Good news! We can do even better than this by deploying mBART 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.

The Jordan Decomposition Theorem: Any such f can be uniquely written (up to an additive constant) as the sum of its monotone up and monotone down components

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

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.*

The Monotone Discovery Strategy with mBART

Key Idea: To discover the monotone decomposition of f, we treat $f(x)$ as embedded in a two-dimensional function

$$f^*(x_1, x_2) = f_{up}(x_1) + f_{down}(x_2).$$

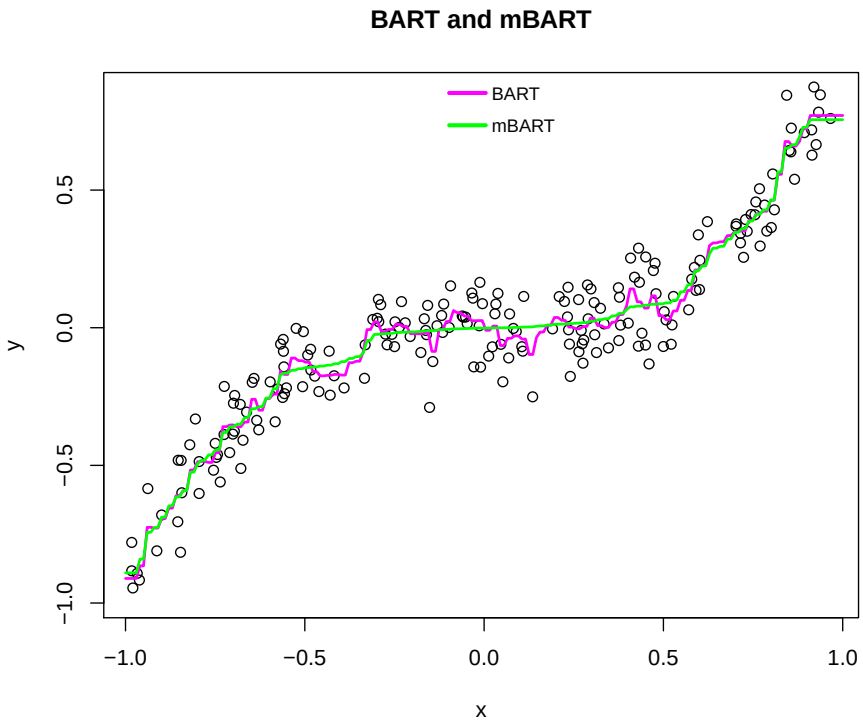
Letting $x_1 = x_2 = x$ be duplicate copies of x, we simply estimate $f^*(x_1, x_2)$ with mBART

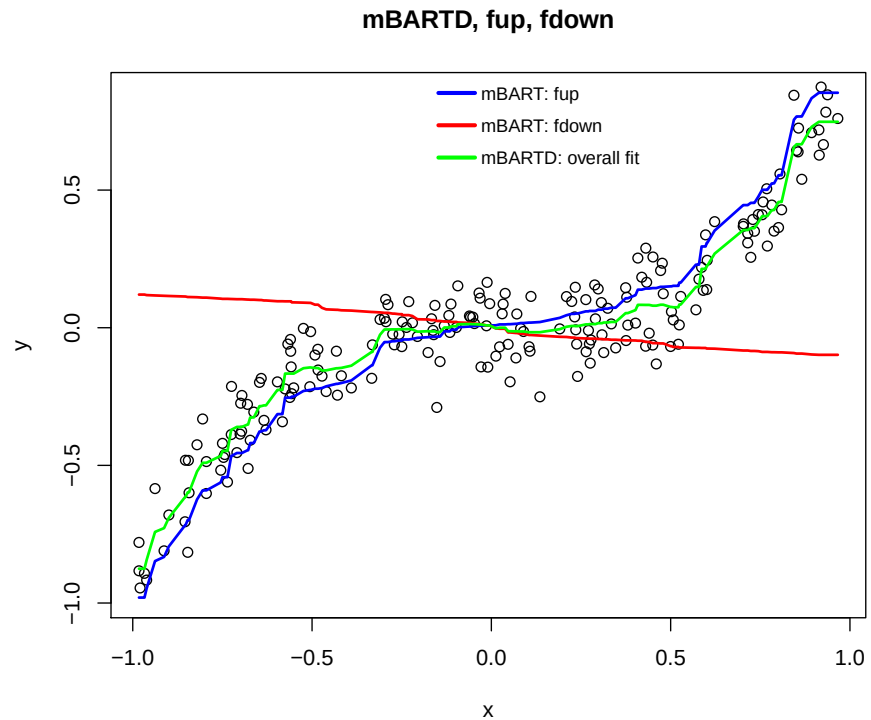
- constrained to be monotone up in the x_1 direction, and
- constrained to be monotone down in the x_2 direction.

Thus, we are estimating the monotone "projections" of $f^*(x_1, x_2)$ along the x_1 and x_2 axes, i.e.

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

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

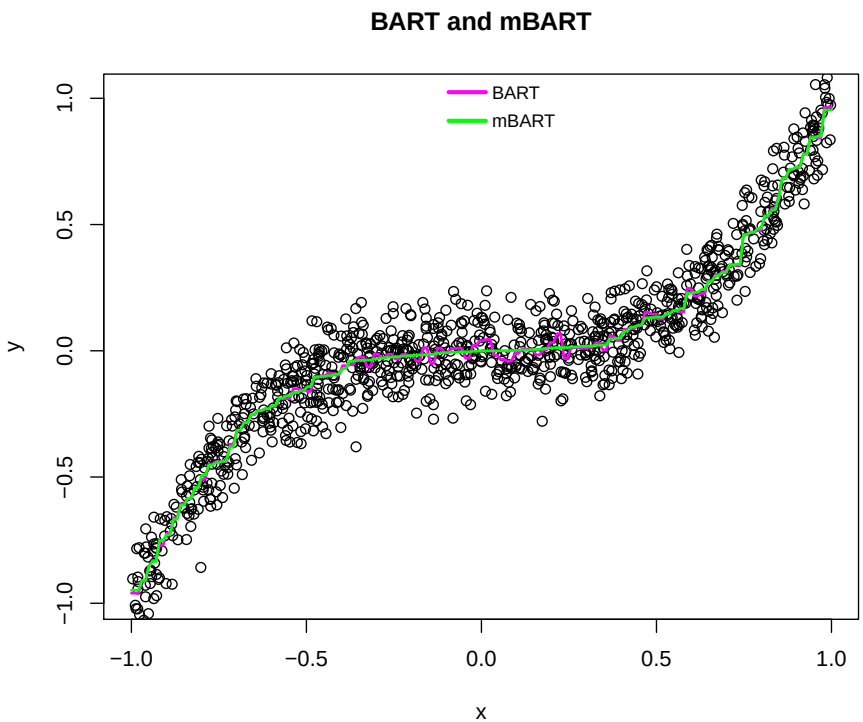


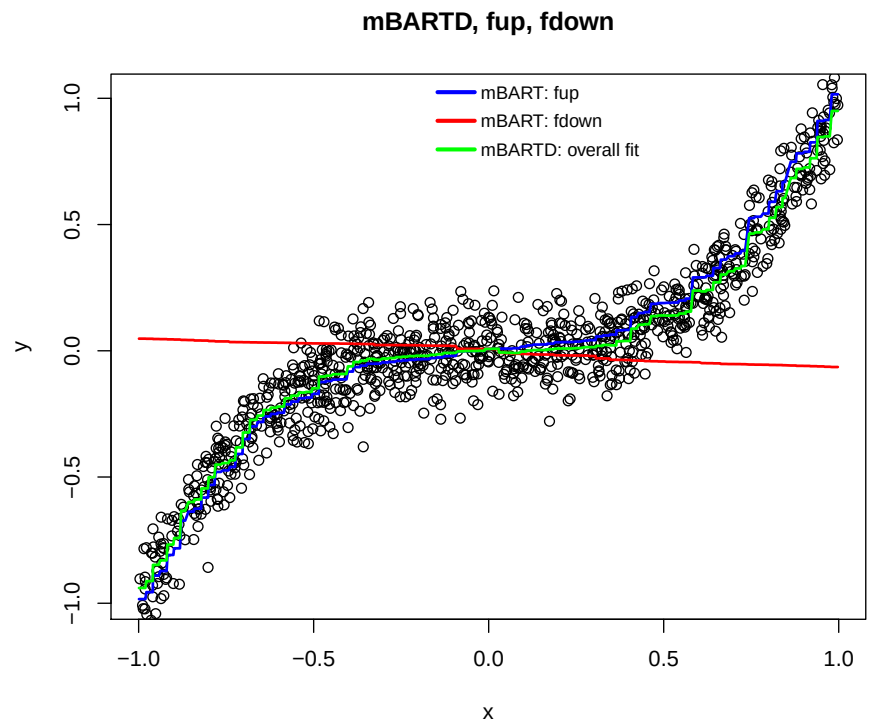


Note that $\hat{f}_{down} \approx 0$ (the red in the right plot), as we would expect when f is monotone up.

Remark: mBARTD $= \hat{f}_{up} + \hat{f}_{down}$ is an alternative estimate of f

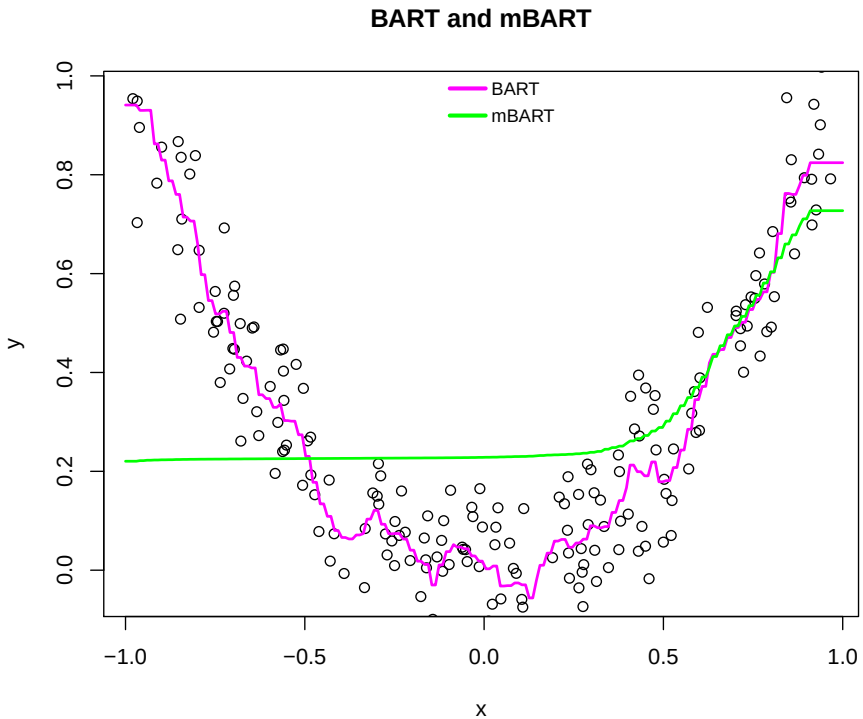
As the sample size is increased from 200 to 1,000, $\hat{f}_{down}$ gets even flatter.

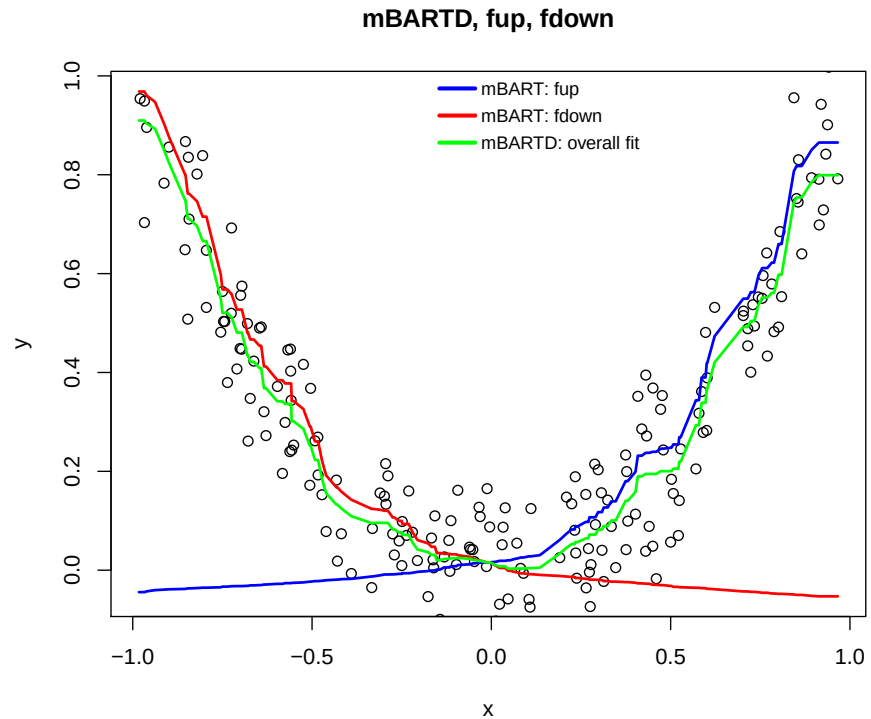




Suggests consistent estimation of the monotone components!!

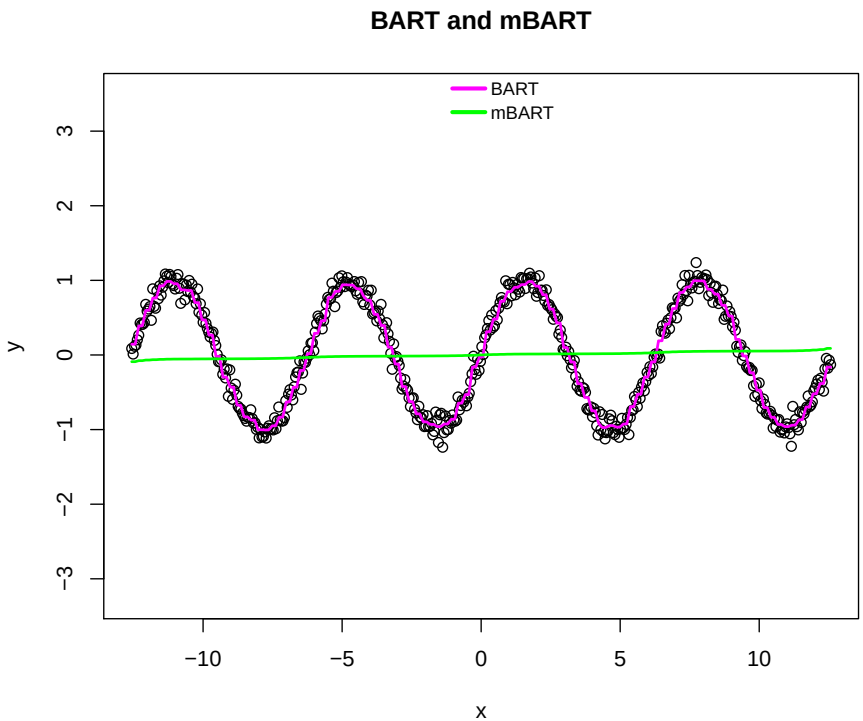
Example: Suppose $Y = x^2 + \epsilon$.

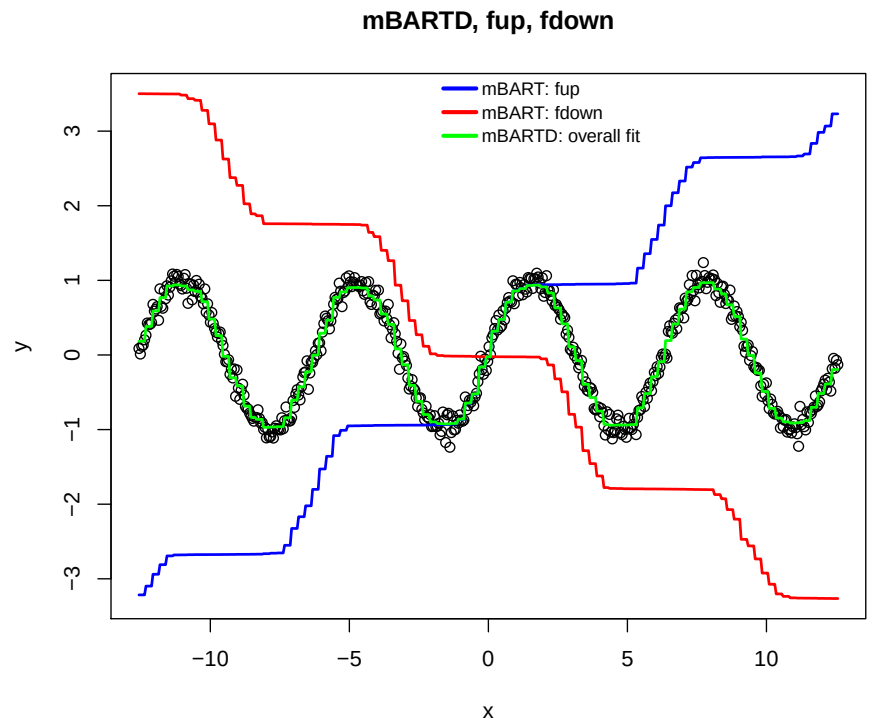




- On the left, BART is good, but simple mBART is not.
- On the right, $\hat{f}_{up}$ and $\hat{f}_{down}$ are spot on.
- And mBARTD $= \hat{f}_{up} + \hat{f}_{down}$ seems even better than BART!

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

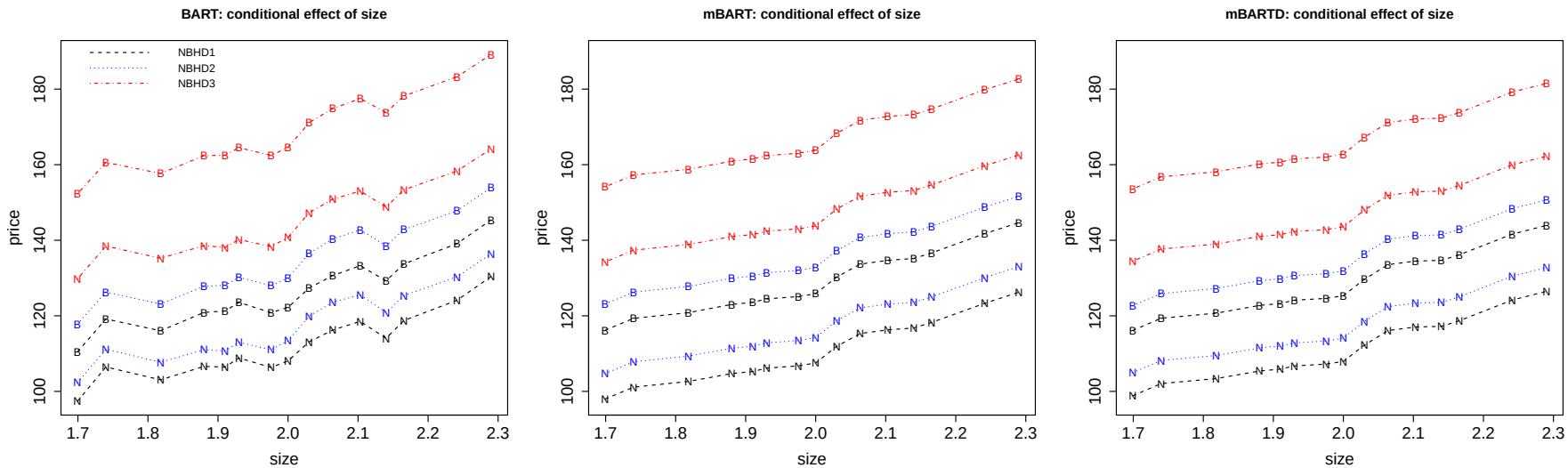
$n = 128$ houses, $y =$ house price ($ thousands),
$x =$ (nbhd (1,2 or 3), size (sq ft thousands), brick (B or N)).

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

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.

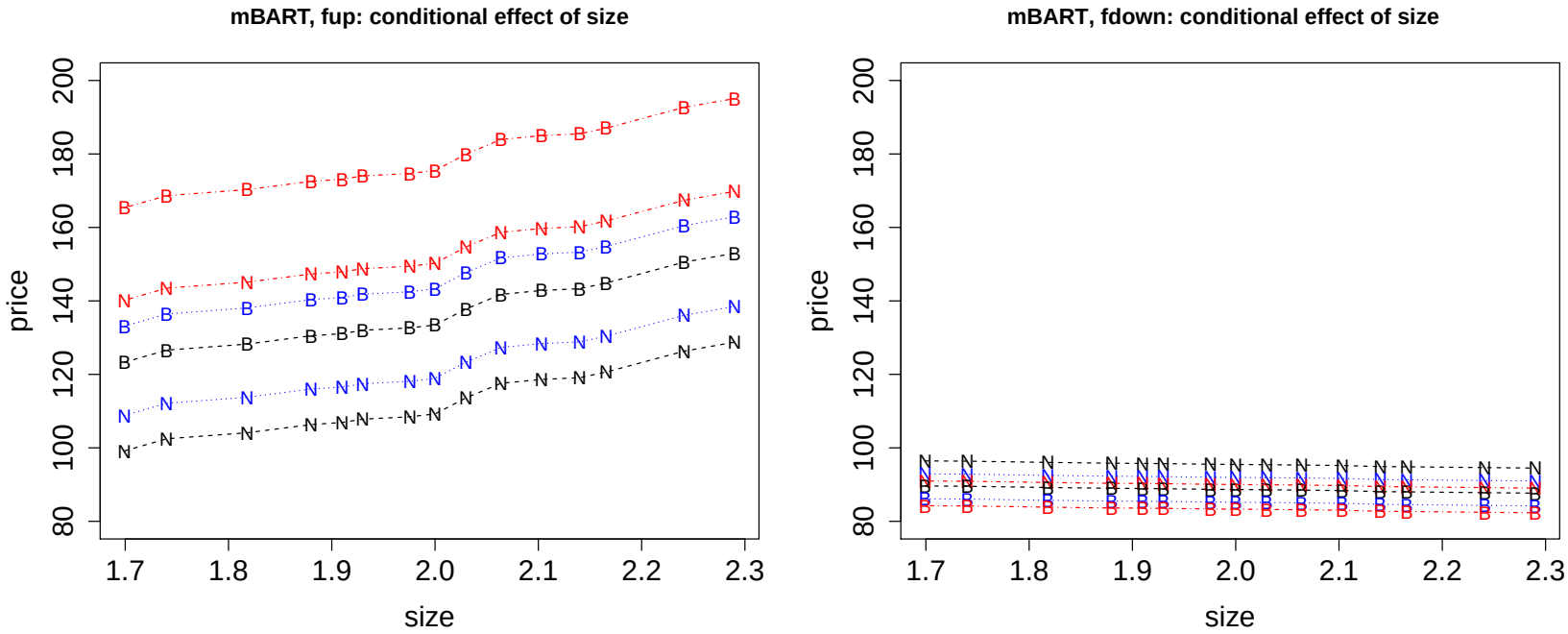
Remark: For the linear model we have to dummy up *nbhd*, but for BART and mBART we can simply leave it as an ordered numerical categorical variable.

Let's first compare BART, mBART (constrained up), and mBARTD to estimate the effect of *size* conditionally on the six possible values of (*nbdh*, *brick*)



Note how mBARTD $= \hat{f}_{up} + \hat{f}_{down}$ adaptively shrinks the estimates towards the mBART estimates.

The full picture emerges from estimates of the effect of *size* via $\hat{f}_{up}$ and $\hat{f}_{down}$ conditionally on the six possible values of $(nbdh, brick)$



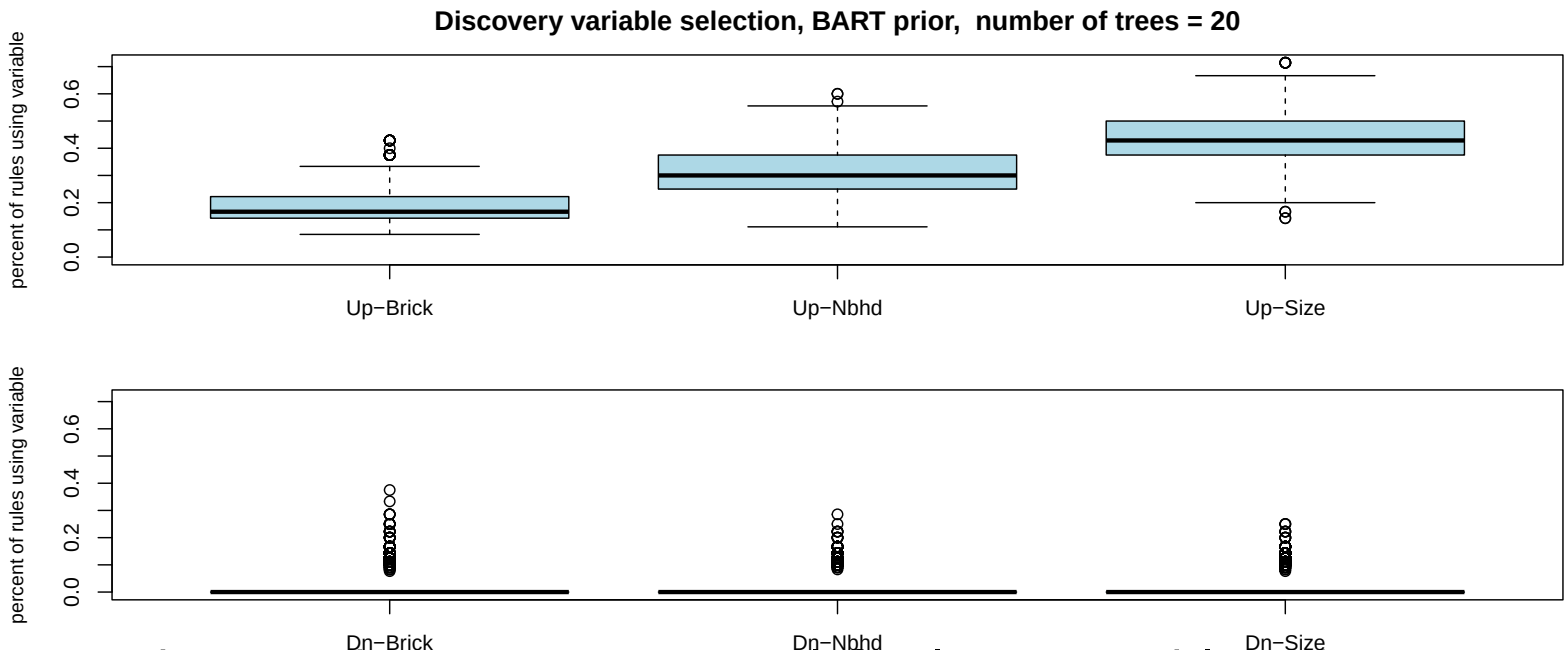
Price is clearly conditionally monotone up in all three variables!

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

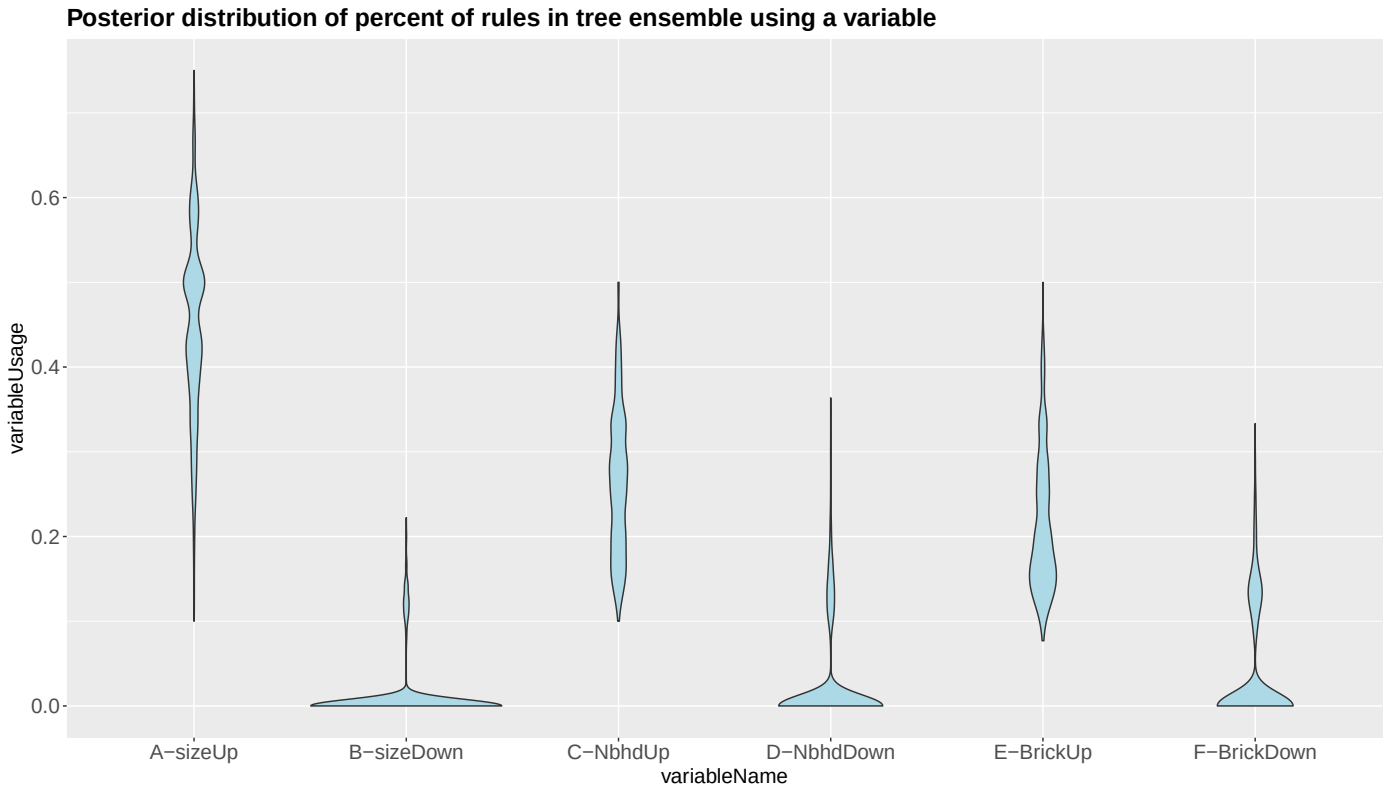
This can all be most conveniently done using the mBART variable importance strategy to gauge the relationships between $y = price$ and $x = (sizeUp, sizeDn, nbhdUp, nbhdDn, brickUp, brickDn)$
$y = price$ and
$x = (UpBrick, UpNbhd, UpSize, DnBrick, DnNbhd, DnSize)$.

$$x = (sizeUp, sizeDn, nbhdUp, nbhdDn, brickUp, brickDn).$$

This frequency-of-use variable importance strategy reveals clearly that *price* is conditionally monotone up in all three variables:



Proceeding in this way is essential for larger problems!

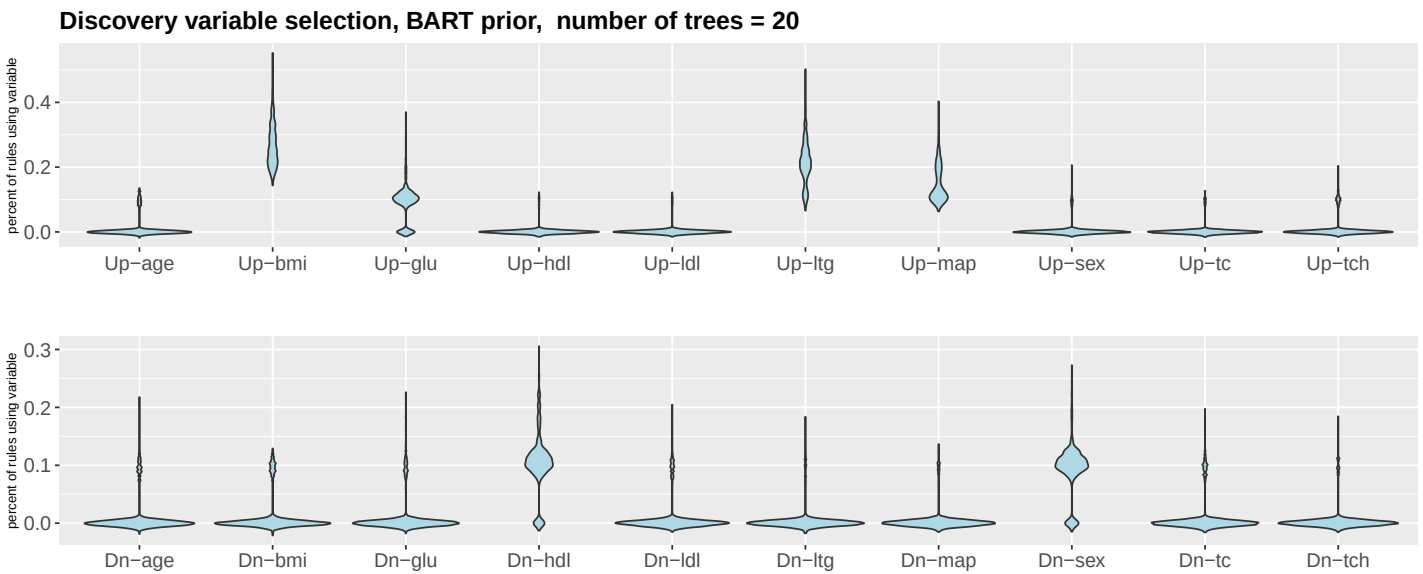

Posterior distribution of percent of rules in tree ensemble using a variable
variableUsage
0.6
0.4
0.2
0.0
A−sizeUp
B−sizeDown
C−NbhdUp
D−NbhdDown
E−BrickUp
F−BrickDown
variableName

Example: The Diabetes Data

Benchmark Dataset used in *Least Angle Regression*
Efron, Hastie, Johnstone, Tibshirani (2004, *AOS*)

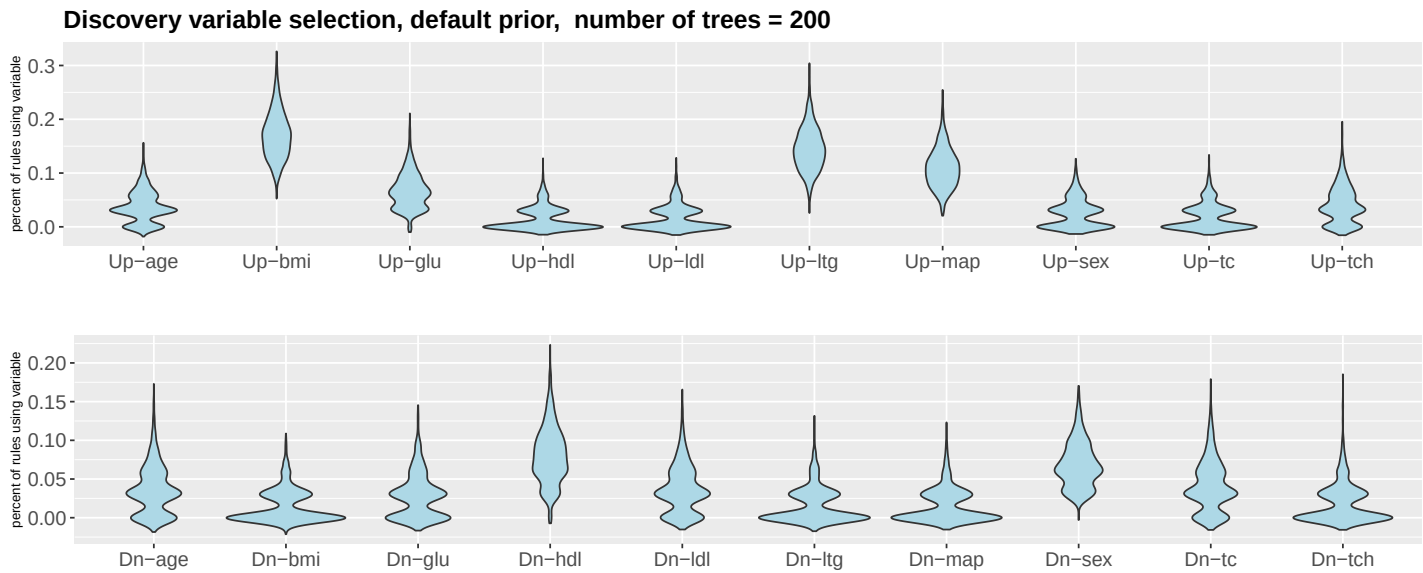
$n = 442$ diabetes patients, $y =$ disease progression measure,
$x =$ (age, bmi, glu, hdl, ldl, ltg, map, sex, tc, tch)

The mBART variable importance strategy identifies six important variables together with the direction of their conditional effects



bmi-U, ltg-U, map-U, glu-U, hdl-D, sex-D.

- Discovery variable importance: put in x and -x.
- BART Prior: put tight prior on σ saying you want the same kind of σ as you got from classic BART.



bmi-U, ltg-U, map-U, glu-U, hdl-D, sex-D.

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.710e-08  2.576e+00   0.000 1.000000
age         -4.768e-01  2.845e+00  -0.168 0.867000
sex         -1.142e+01  2.915e+00  -3.917 0.000104 ***
bmi          2.475e+01  3.168e+00   7.813 4.30e-14 ***
map          1.545e+01  3.115e+00   4.958 1.02e-06 ***
tc          -3.772e+01  1.984e+01  -1.901 0.057947 .
ldl          2.270e+01  1.614e+01   1.406 0.160389
hdl          4.812e+00  1.012e+01   0.475 0.634720
tch          8.432e+00  7.689e+00   1.097 0.273456
ltg          3.578e+01  8.186e+00   4.370 1.56e-05 ***
glu          3.220e+00  3.142e+00   1.025 0.305998
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 54.15 on 431 degrees of freedom
Multiple R-squared:  0.5177,Adjusted R-squared:  0.5066
F-statistic: 46.27 on 10 and 431 DF,  p-value: < 2.2e-16
```

Not the same !!!!

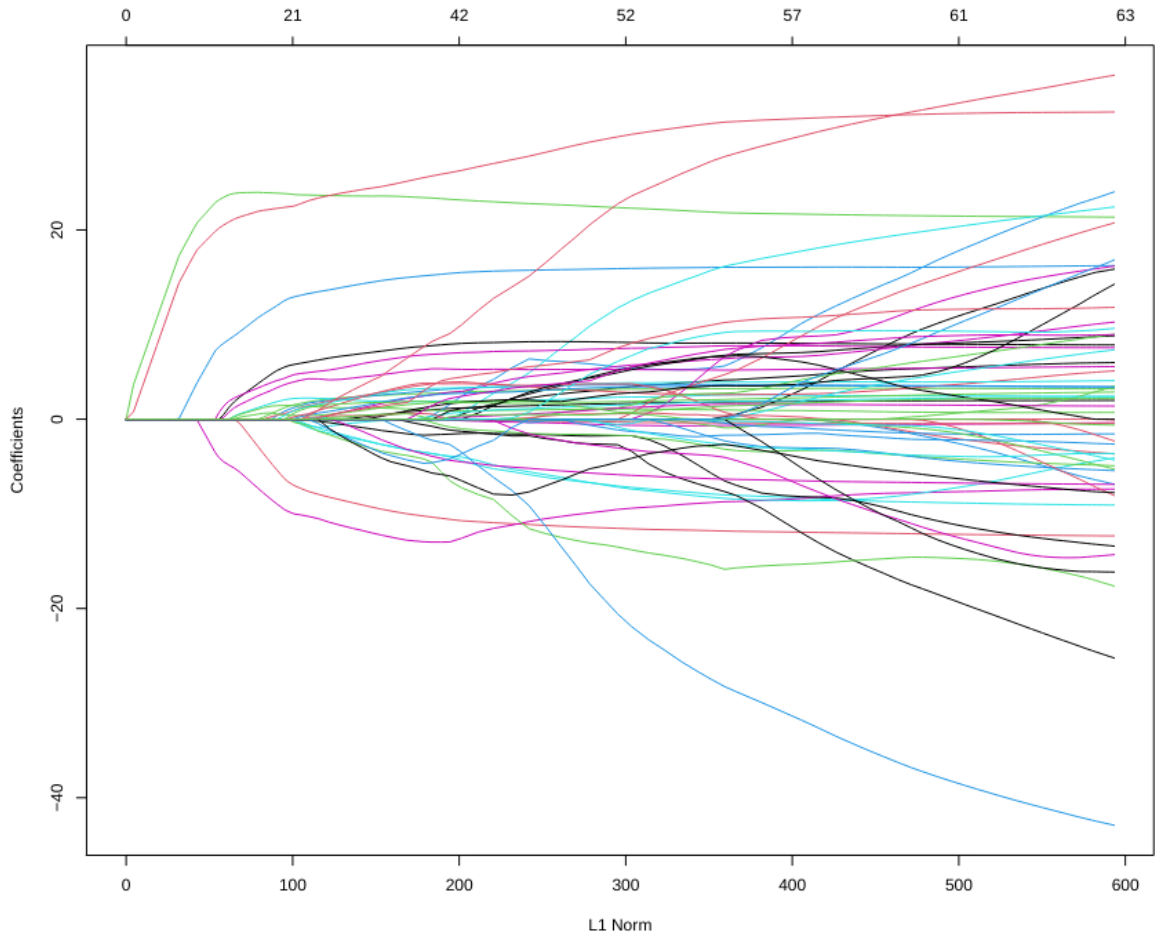
which one is more useful !!!????

How about this ????

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.710e-08  2.532e+00   0.000   1.0000
age          2.415e+00  3.120e+00   0.774   0.4393
sex         -1.273e+01  3.108e+00  -4.096 5.15e-05 ***
bmi          2.194e+01  4.029e+00   5.446 9.32e-08 ***
map          1.633e+01  3.450e+00   4.734 3.13e-06 ***
tc          -1.717e+02  2.885e+03  -0.060   0.9526
ldl          1.444e+02  2.535e+03   0.057   0.9546
hdl          5.263e+01  1.078e+03   0.049   0.9611
tch          3.568e+00  1.313e+01   0.272   0.7860
ltg          8.715e+01  9.483e+02   0.092   0.9268
glu          2.988e+00  3.352e+00   0.891   0.3733
age.2        3.223e+00  3.308e+00   0.974   0.3305
bmi.2        2.183e+00  3.966e+00   0.550   0.5823
map.2       -4.028e-01  3.412e+00  -0.118   0.9061
tc.2         3.175e+02  3.361e+02   0.945   0.3455
ldl.2        1.706e+02  2.536e+02   0.673   0.5016
hdl.2        8.246e+01  7.574e+01   1.089   0.2770
tch.2        3.683e+01  2.890e+01   1.274   0.2034
ltg.2        6.913e+01  8.239e+01   0.839   0.4019
glu.2        5.436e+00  4.482e+00   1.213   0.2260
age.sex      7.080e+00  3.496e+00   2.025   0.0435 *
age.bmi     -8.596e-01  3.791e+00  -0.227   0.8208
age.map      8.825e-01  3.633e+00   0.243   0.8082
age.tc      -7.566e+00  2.939e+01  -0.257   0.7969
age.ldl     -3.204e+00  2.355e+01  -0.136   0.8919
age.hdl      9.964e+00  1.336e+01   0.746   0.4563
age.tch      8.808e+00  1.002e+01   0.879   0.3798
age.ltg      5.937e+00  1.066e+01   0.557   0.5778
age.glu      2.980e+00  3.827e+00   0.779   0.4367
sex.bmi      3.077e+00  3.710e+00   0.829   0.4074
sex.map      4.213e+00  3.559e+00   1.184   0.2373
sex.tc       2.065e+01  2.813e+01   0.734   0.4634
sex.ldl     -1.680e+01  2.233e+01  -0.752   0.4523
sex.hdl     -5.940e+00  1.304e+01  -0.455   0.6491
sex.tch     -6.249e+00  9.510e+00  -0.657   0.5115
sex.ltg     -5.666e+00  1.079e+01  -0.525   0.5996
sex.glu      2.179e+00  3.507e+00   0.621   0.5348
bmi.map      7.368e+00  4.111e+00   1.792   0.0739 .
bmi.tc      -1.438e+01  3.181e+01  -0.452   0.6514
bmi.ldl      1.150e+01  2.672e+01   0.431   0.6670
bmi.hdl      5.807e+00  1.571e+01   0.370   0.7118
bmi.tch     -1.593e+00  1.099e+01  -0.145   0.8849
bmi.ltg      5.461e+00  1.219e+01   0.448   0.6544
bmi.glu      1.113e+00  4.335e+00   0.257   0.7975
map.tc       2.278e+01  3.249e+01   0.701   0.4837
map.ldl     -1.556e+01  2.735e+01  -0.569   0.5698
map.hdl     -8.919e+00  1.474e+01  -0.605   0.5455
map.tch     -2.776e+00  9.457e+00  -0.294   0.7693
map.ltg     -7.371e+00  1.295e+01  -0.569   0.5696
map.glu     -6.356e+00  4.348e+00  -1.462   0.1447
tc.ldl      -4.435e+02  5.605e+02  -0.791   0.4294
tc.hdl      -1.872e+02  1.817e+02  -1.030   0.3036
tc.tch      -1.050e+02  8.390e+01  -1.252   0.2113
tc.ltg      -1.810e+02  6.270e+02  -0.289   0.7730
tc.glu      -8.395e+00  2.836e+01  -0.296   0.7673
ldl.hdl      1.258e+02  1.508e+02   0.835   0.4045
ldl.tch      5.747e+01  7.002e+01   0.821   0.4124
ldl.ltg      1.320e+02  5.219e+02   0.253   0.8004
ldl.glu      4.077e+00  2.405e+01   0.170   0.8655
hdl.tch      5.659e+01  4.773e+01   1.186   0.2365
hdl.ltg      6.988e+01  2.195e+02   0.318   0.7504
hdl.glu      1.036e+01  1.413e+01   0.733   0.4640
tch.ltg      1.856e+01  2.975e+01   0.624   0.5330
tch.glu      1.122e+01  1.119e+01   1.003   0.3167
ltg.glu      3.977e+00  1.261e+01   0.316   0.7525
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 53.23 on 377 degrees of freedom
Multiple R-squared:  0.5924,Adjusted R-squared:  0.5233
F-statistic: 8.563 on 64 and 377 DF,  p-value: < 2.2e-16
```

How about this ????

Here is the Lasso coefficient plot.



Boosting or RF or Deep Learning with SHAP values ???!!!

Still more quesions than answers, *but I would definitely use mBARTD !!!!!!*