Multidimensional Monotonicity Discovery with mBART

Robert McCulloch
Milwaukee, April 2026
robert.e.mcculloch@gmail.com

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

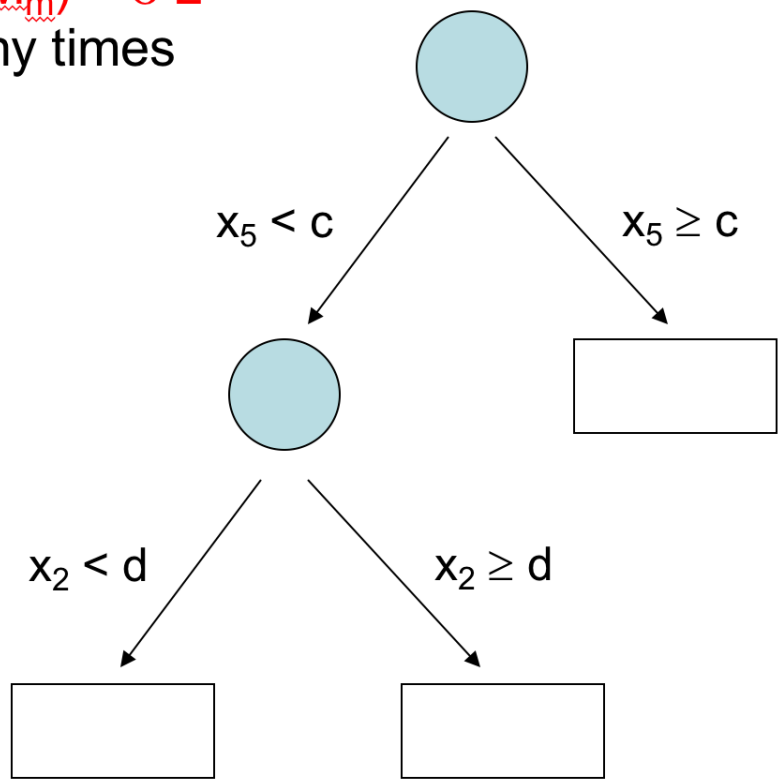
Measuring Variable Importance with BART

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.

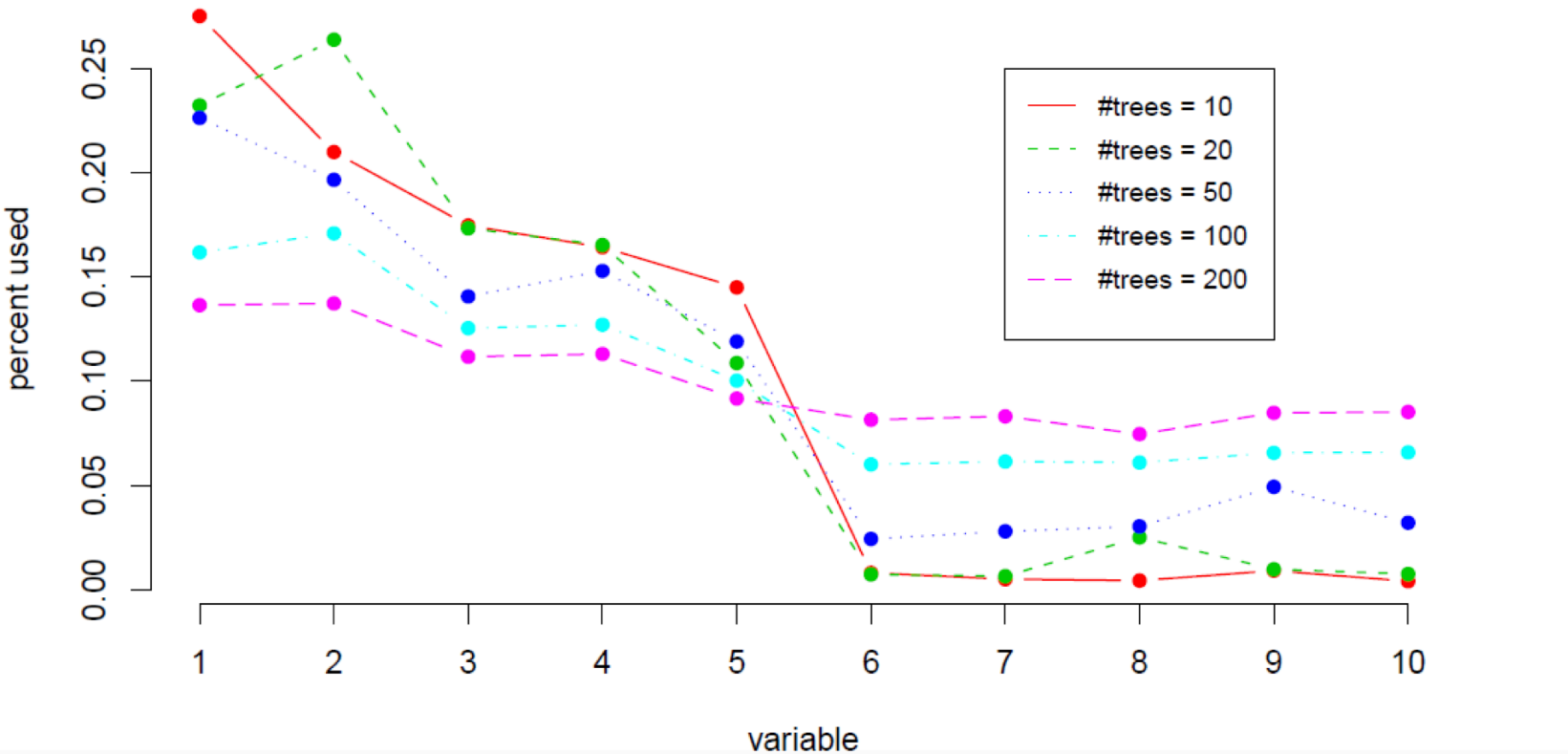


Example: The Friedman Test Function

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



Monotone BART - mBART

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

The Key Idea:

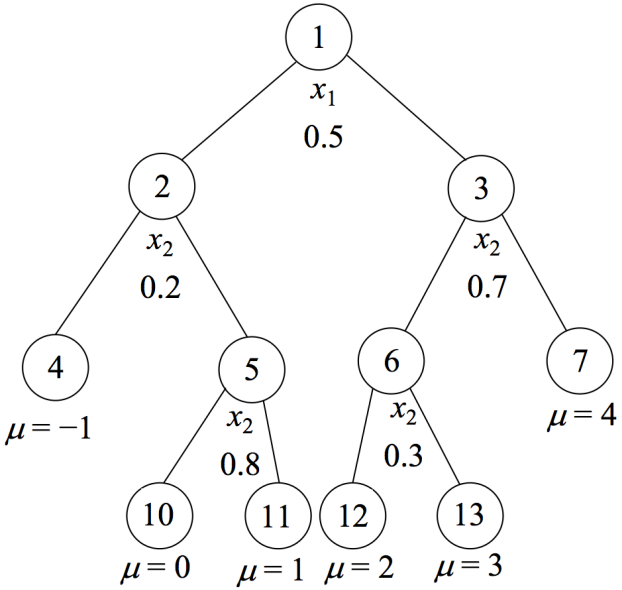
BART approximates a function by a sum of tree functions

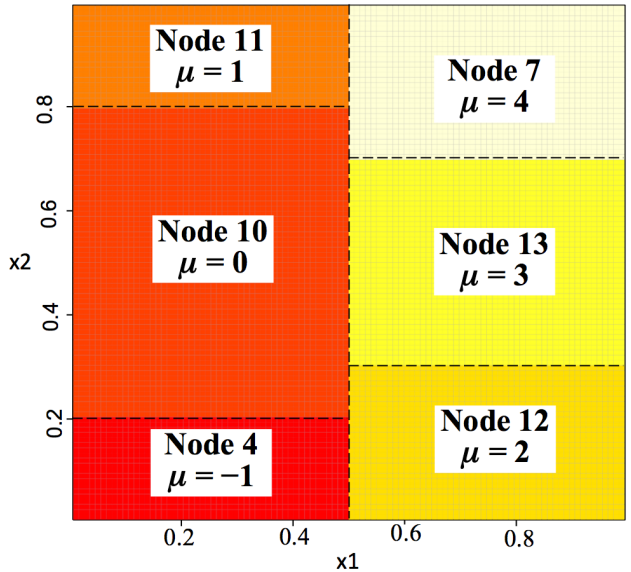
mBART approximates a monotone function by a sum of monotone tree functions

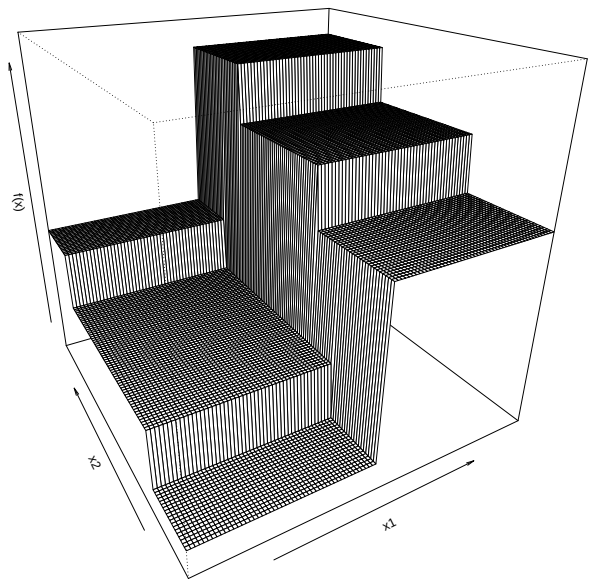
This works because of the obvious fact:

The sum of monotone functions yields a monotone function

An Example of a Monotone Tree Function

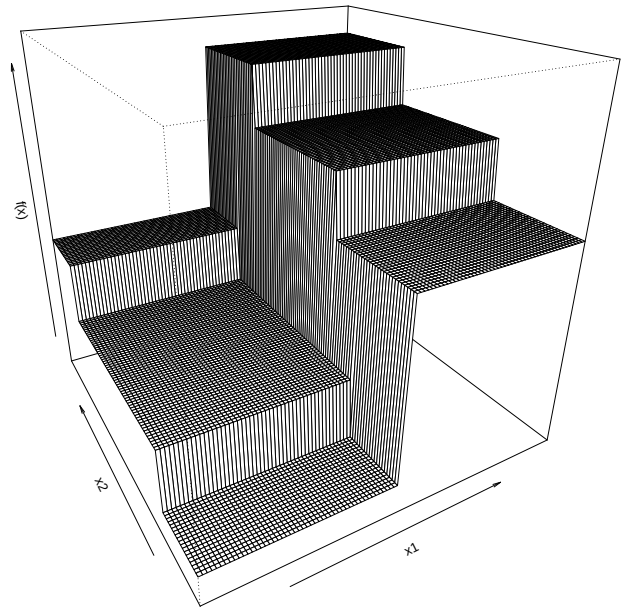






Three different views of a bivariate monotone tree.

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

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

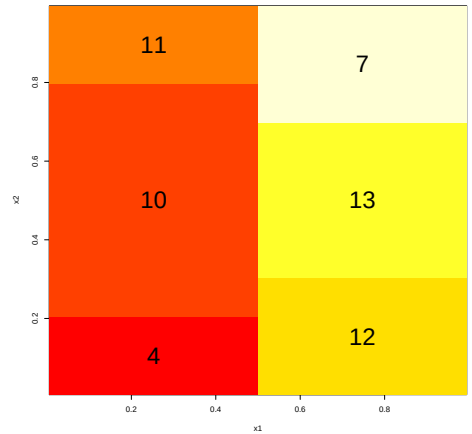
Let $S = \{\theta : \text{each } (T_j, M_j) \text{ is monotone in a desired subset of } x_i's\}$

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.

Comments on Algorithm

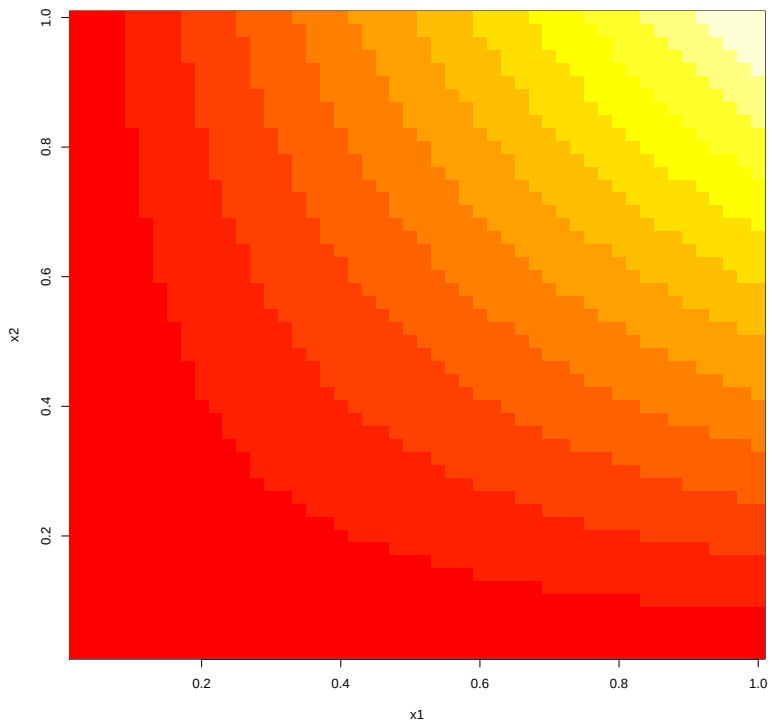
- MCMC works on a single tree at a time.
- M_{New} is the small set of μ's associated with a proposed change in the tree.
- We can figure out the contraints on the M_{New} given the part of the tree we are not touching and the associated M_{Old}.
- In constrained problems where M_{New} cannot be analytically integrated out, the small size of M_{New} makes numerical approximation of the integral by discretization feasible.

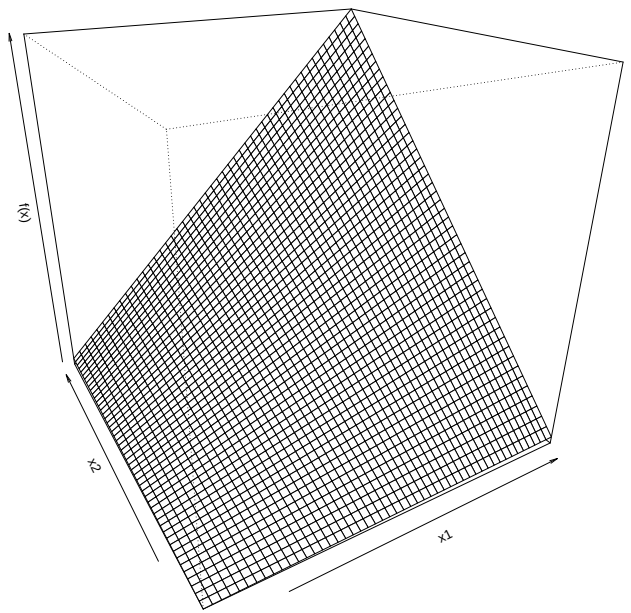
Example: Product of two x's

Let's consider a very simple simulated monotone example:

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

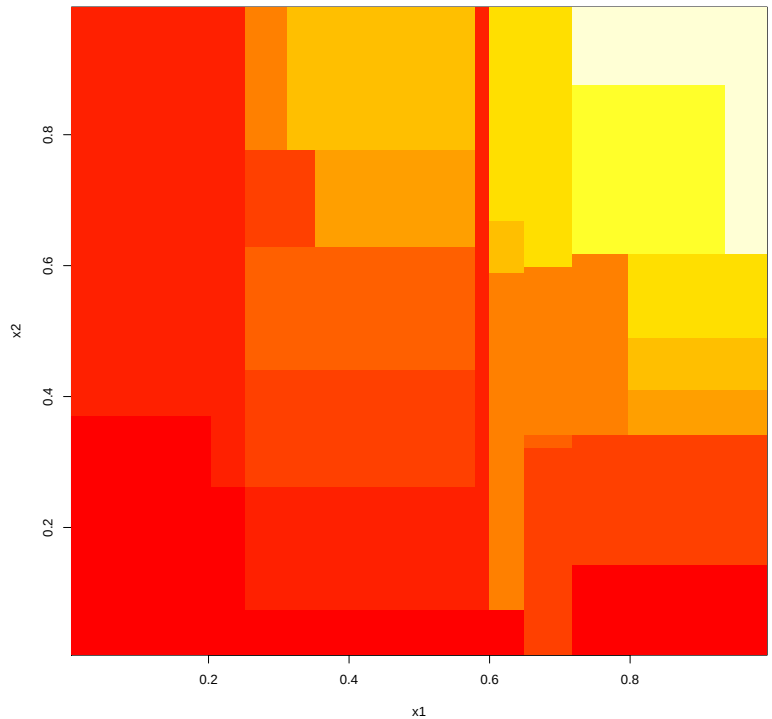
Here is the plot of the true function $f(x_1, x_2) = x_1 \, x_2$

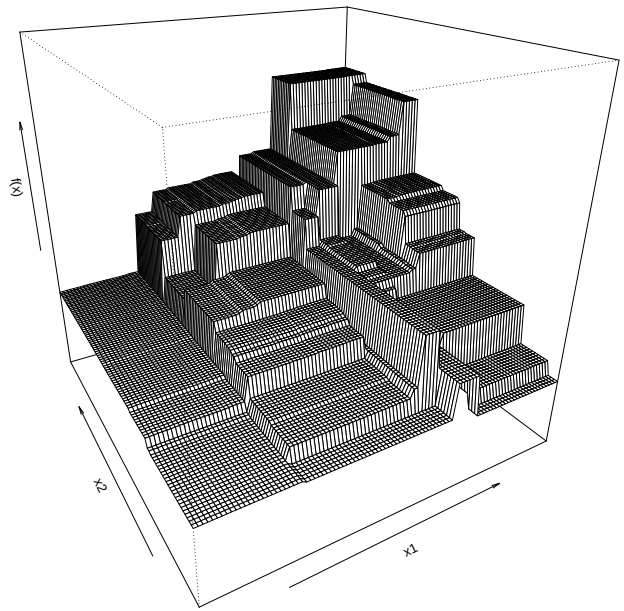




First we try a single (just one tree), unconstrained tree model.

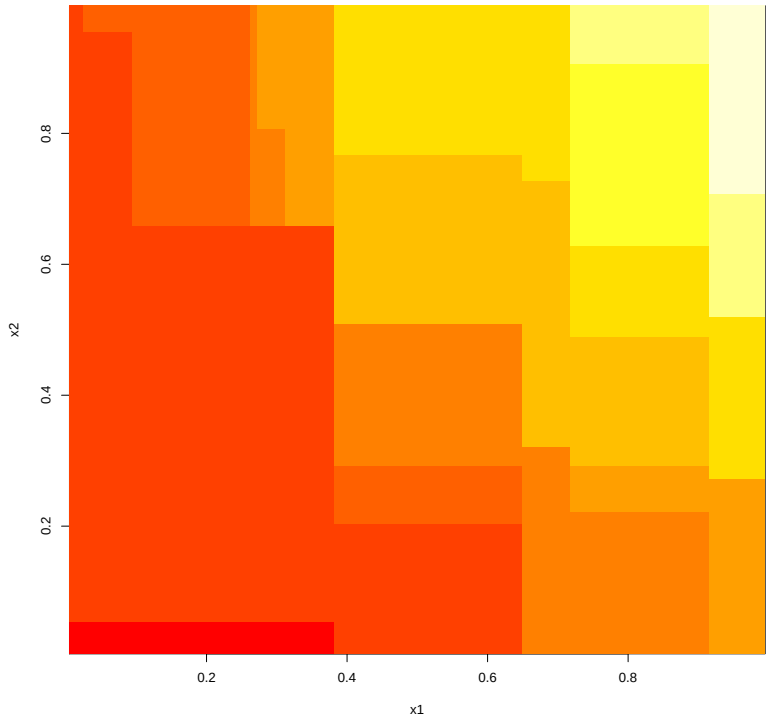
Here is the graph of the fit.

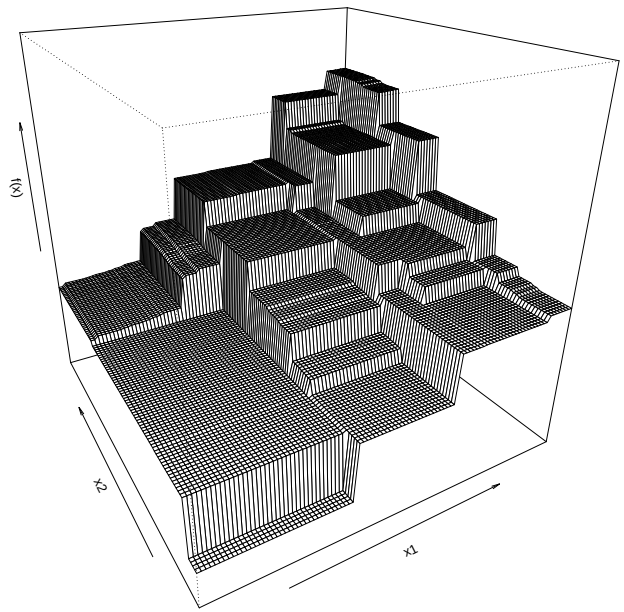




The fit is not terrible, but there are some aspects of the fit which violate monotonicity.

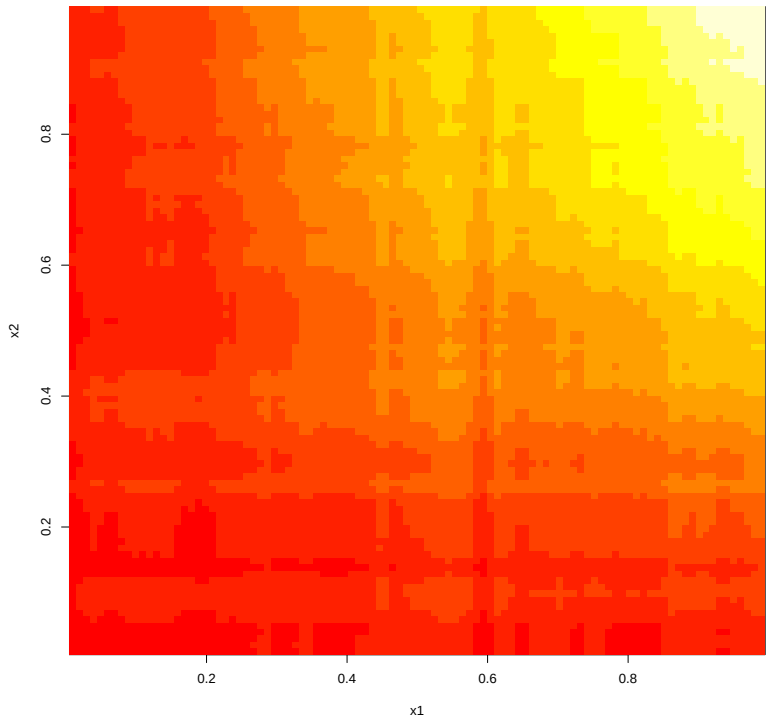
Here is the graph of the fit with the monotone constraint:

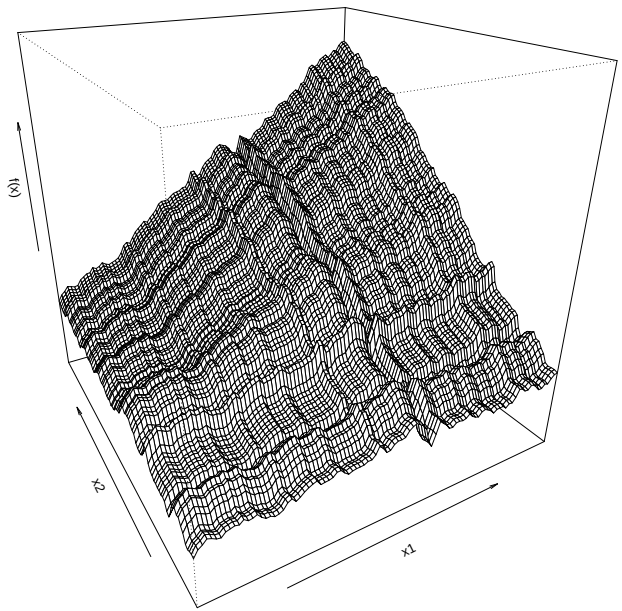




We see that our fit is monotonic, and more representative of the true f.

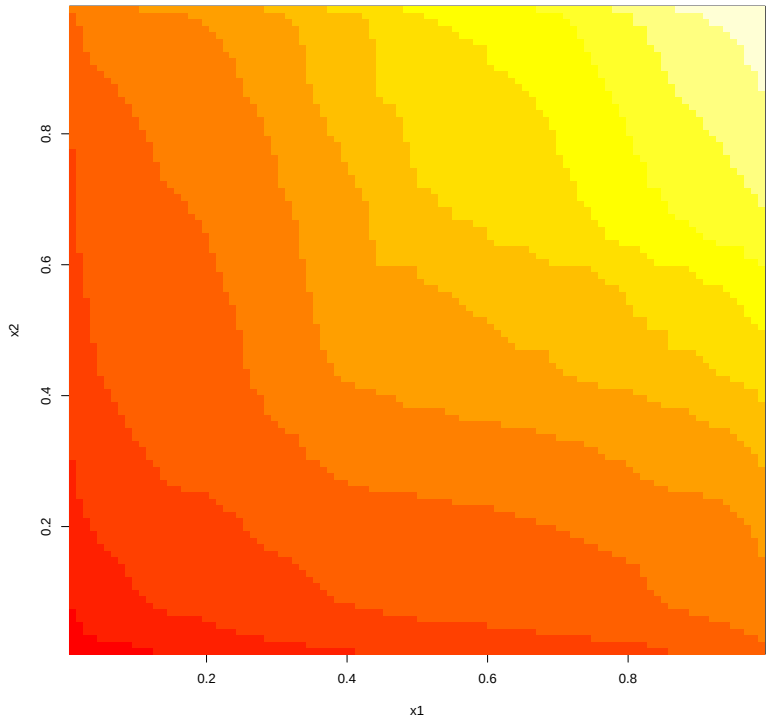
Here is the unconstrained BART fit:

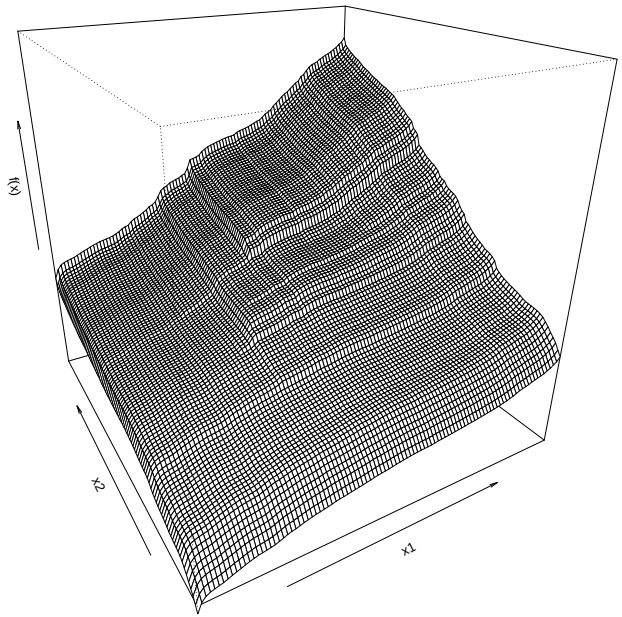




Much better (of course) but not monotone!

And, finally, the constrained BART fit:



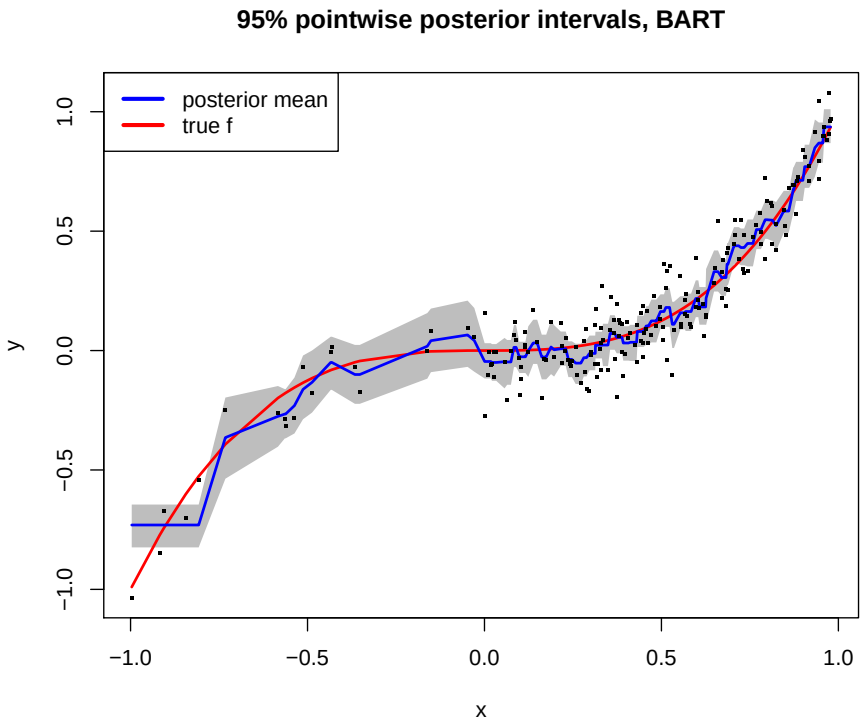


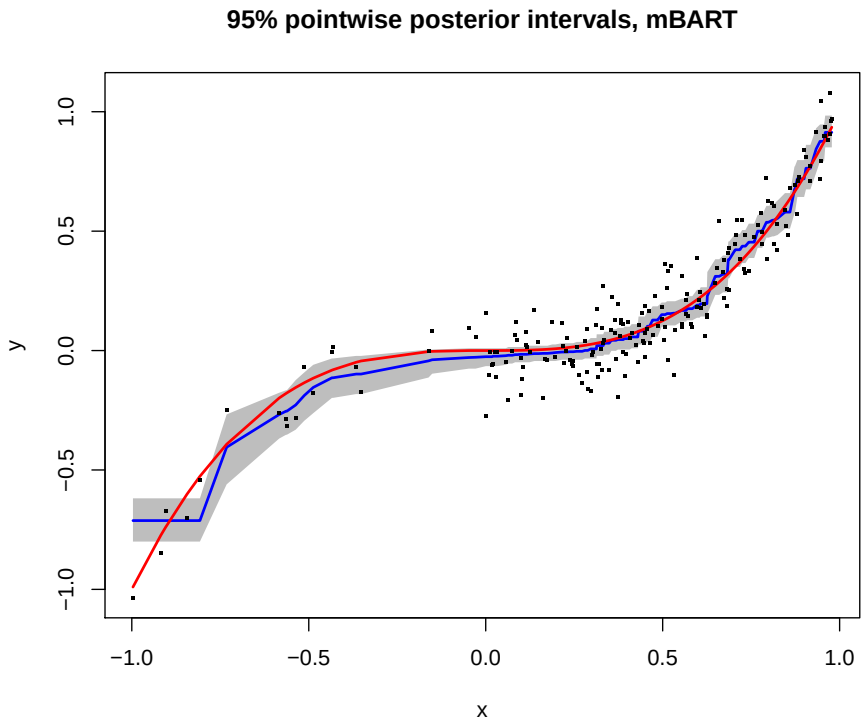
Not Bad!

Same method works with any number of x's!

Automatic Uncertainty Quantification

1-dimensional simulated example





mBART intervals are tighter!

Example: RMSE Reduction by Monotone Regularization

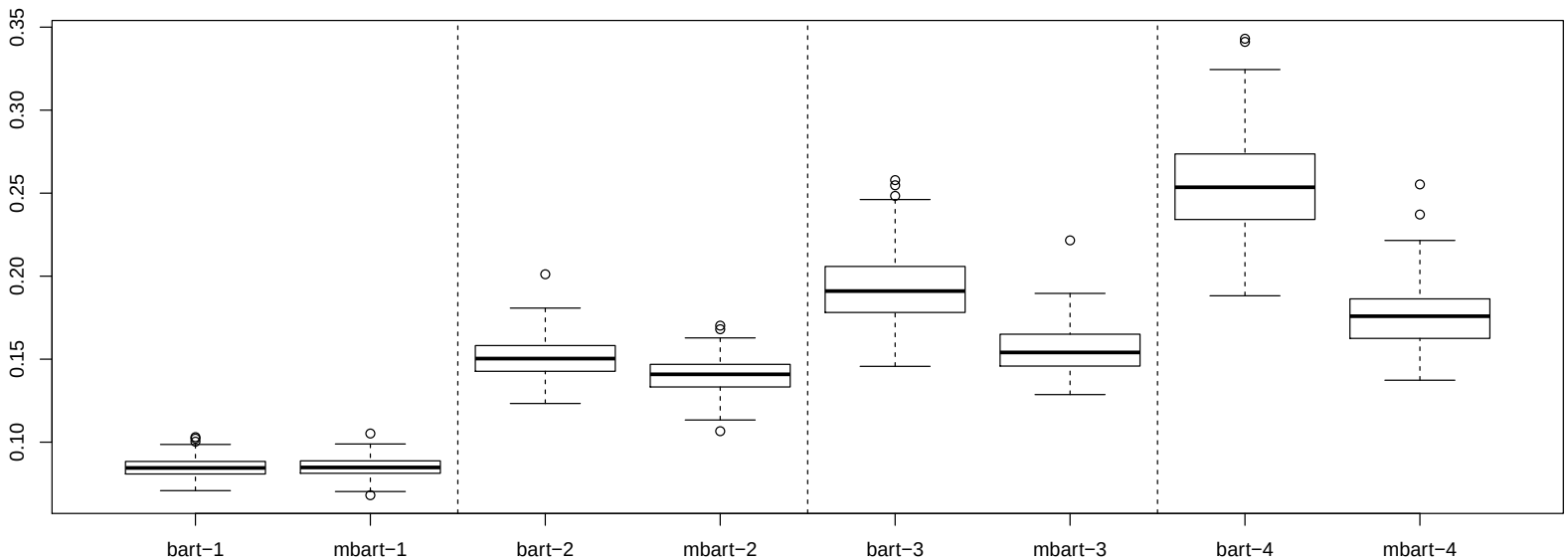
$$Y = x_1 x_2^2 + x_3 x_4^3 + x_5 + \epsilon,$$

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

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%



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

Discovering Monotonicity with mBART

If you have prior information about monotonicity, incorporating that information into the prior can improve predictions and reduce uncertainty.

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

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

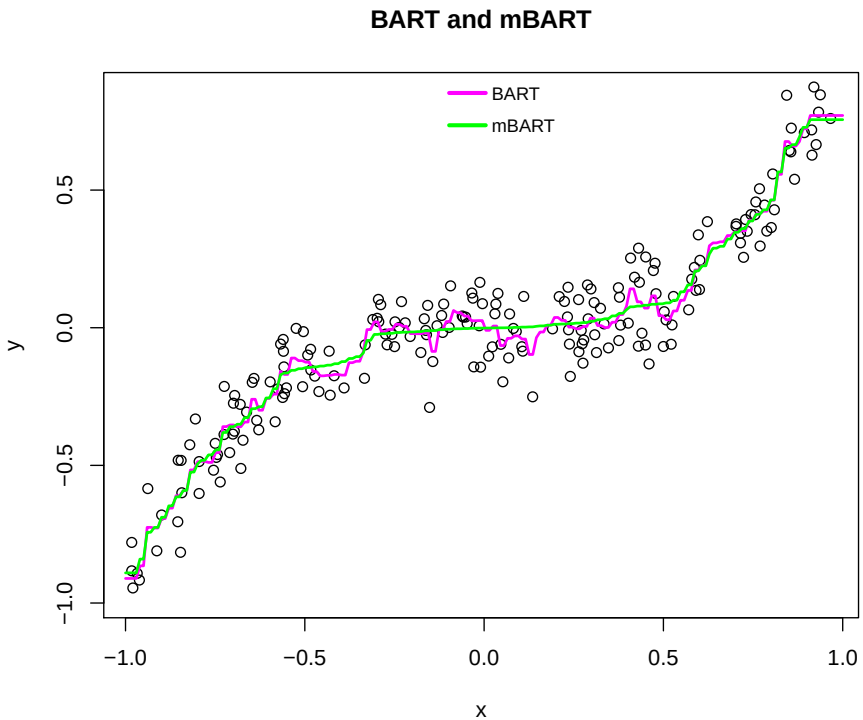
Letting $x_1 = x_2 = x$ be duplicate copies of x, we simply estimate $f^*(x_1, x_2)$ with mBART

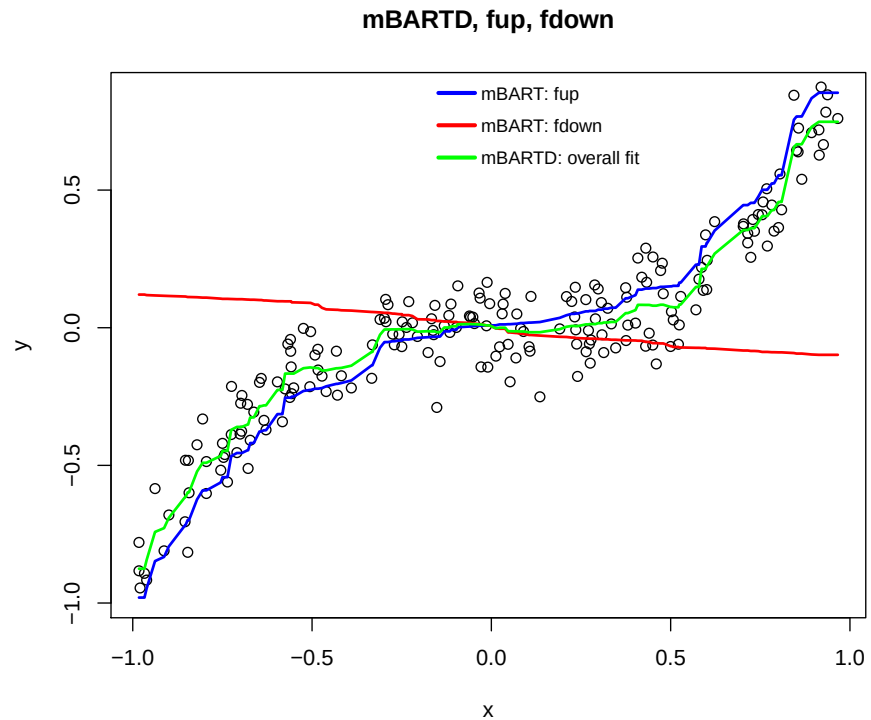
- constrained to be monotone up in the x_1 direction, and
- constrained to be monotone down in the x_2 direction.

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 $Y = x^3 + \epsilon$.

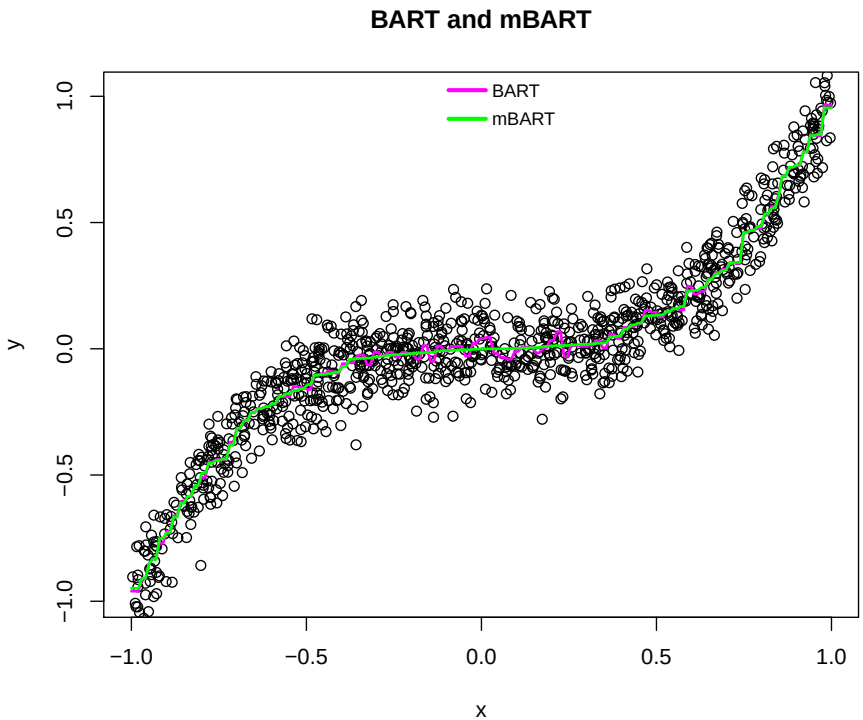


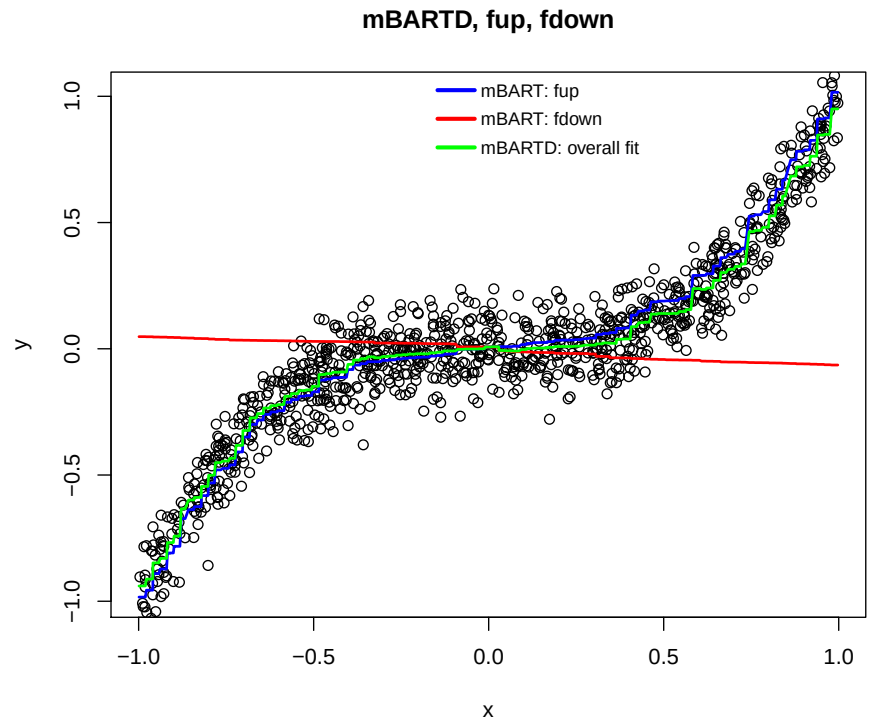


Note that $\hat{f}_{down} \approx 0$ (the red in the right plot), as we would expect when f is monotone up.

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

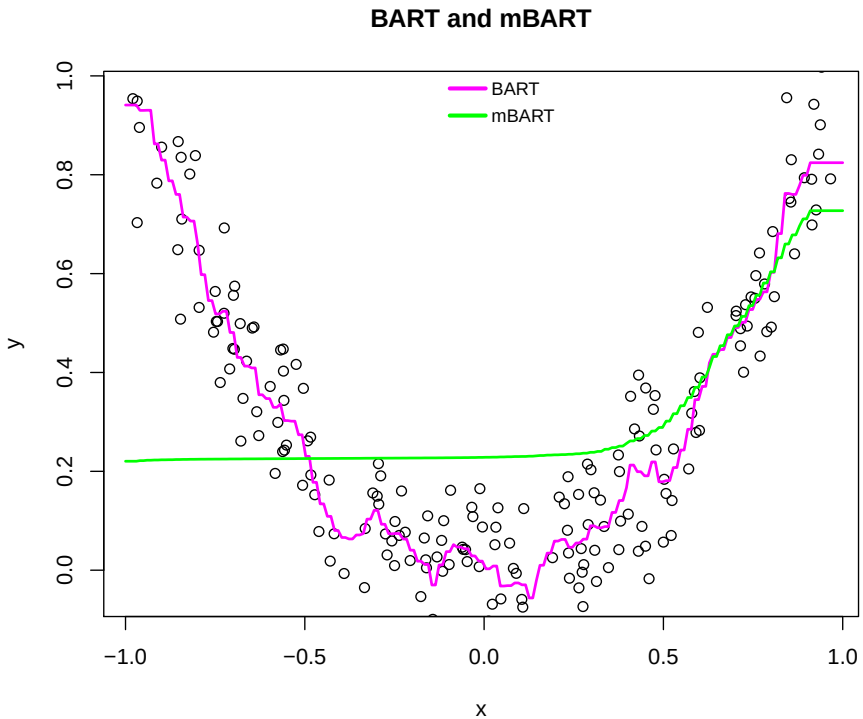
As the sample size is increased from 200 to 1,000, $\hat{f}_{down}$ gets even flatter.

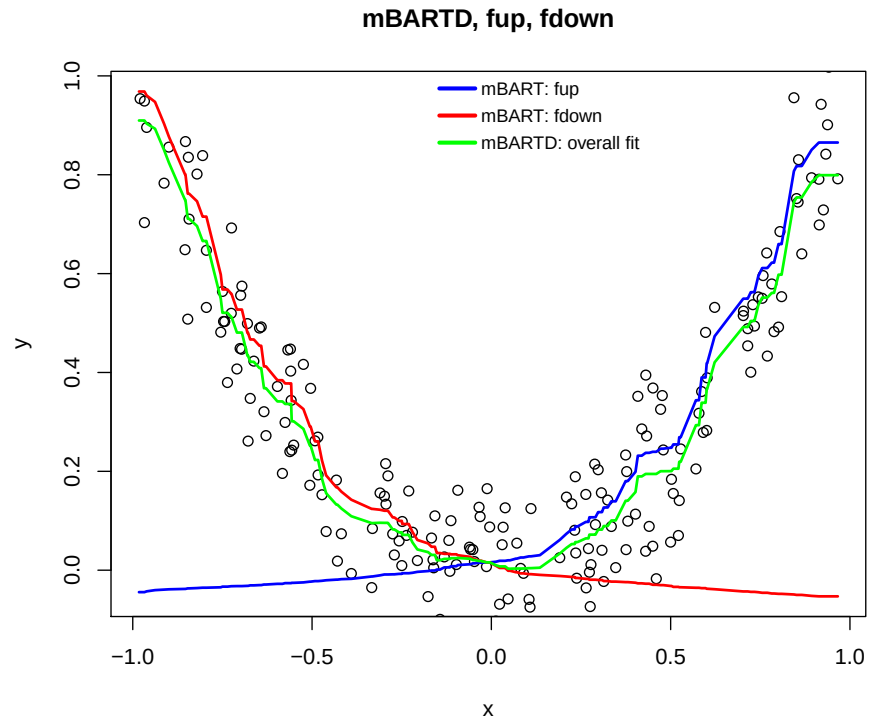




Suggests consistent estimation of the monotone components!!

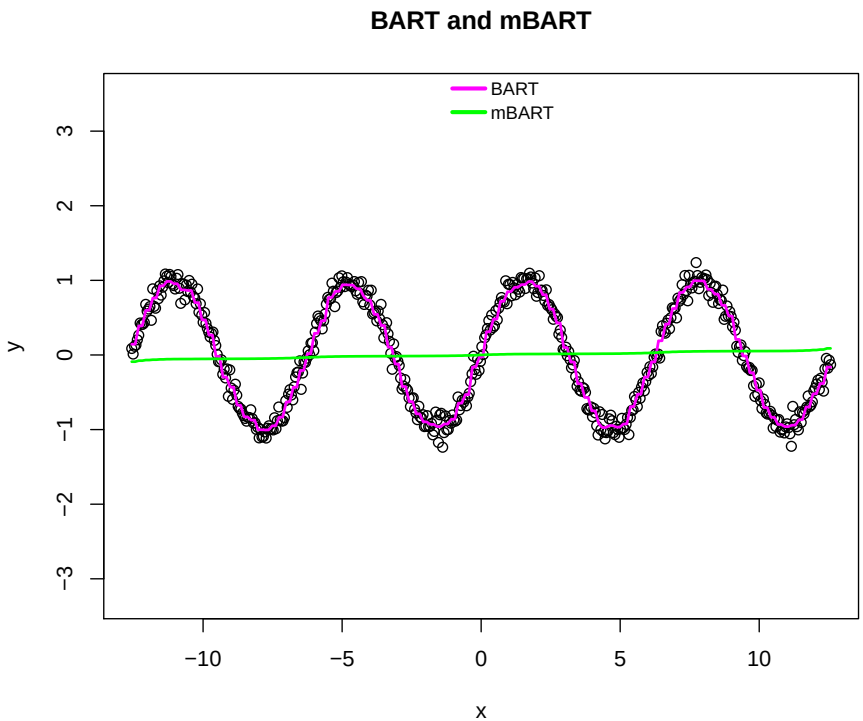
Example: Suppose $Y = x^2 + \epsilon$.

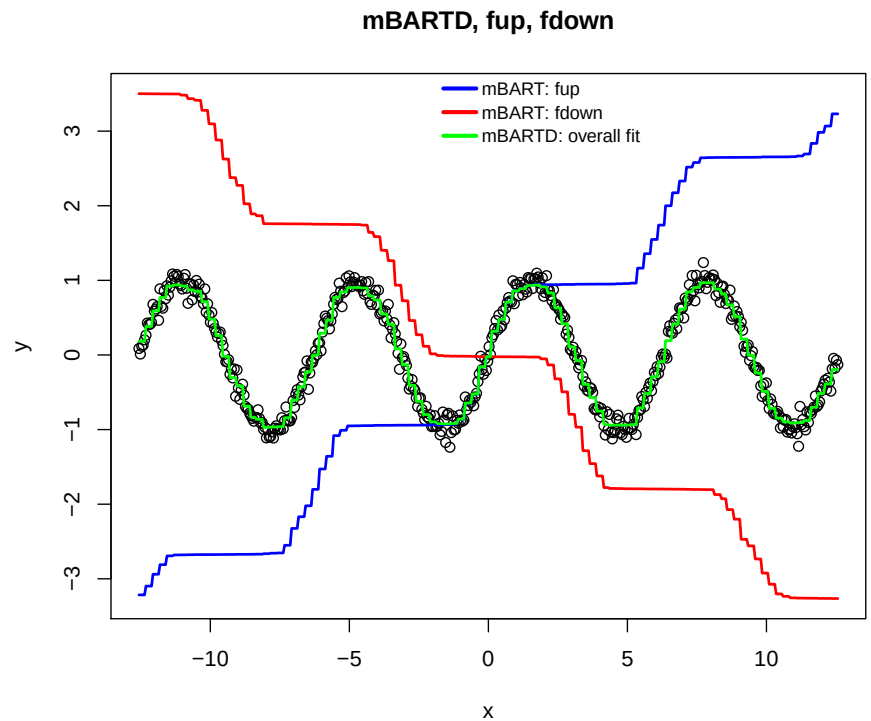




- On the left, BART is good, but simple mBART is not.
- On the right, $\hat{f}_{up}$ and $\hat{f}_{down}$ are spot on.
- And mBARTD $= \hat{f}_{up} + \hat{f}_{down}$ seems even better than BART!

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

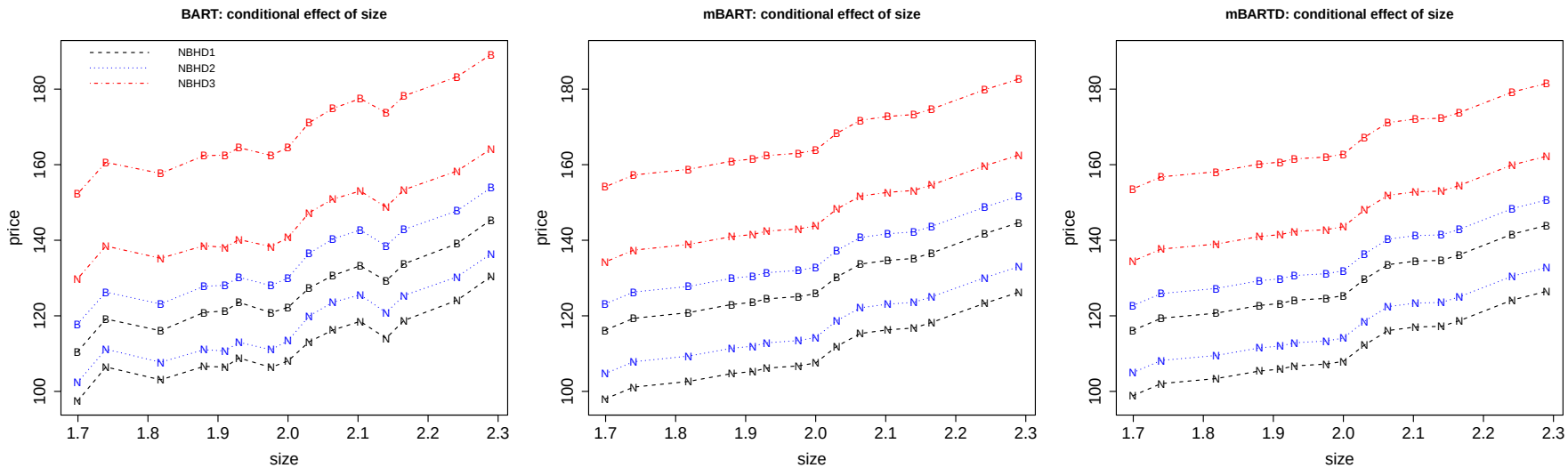
$n = 128$ houses, $y =$ house price ($ thousands),
$x =$ (nbhd (1,2 or 3), size (sq ft thousands), brick (B or N)).

```
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.

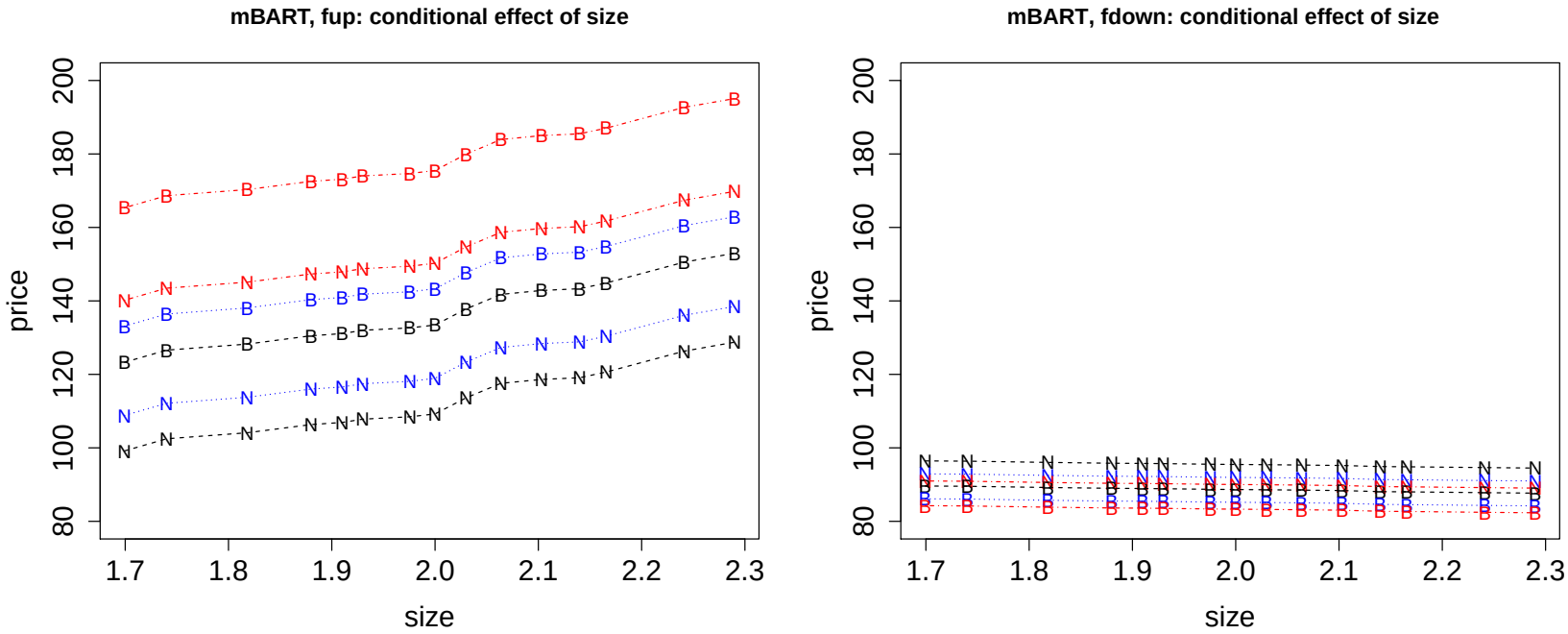
Remark: For the linear model we have to dummy up *nbhd*, but for BART and mBART we can simply leave it as an ordered numerical categorical variable.

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!

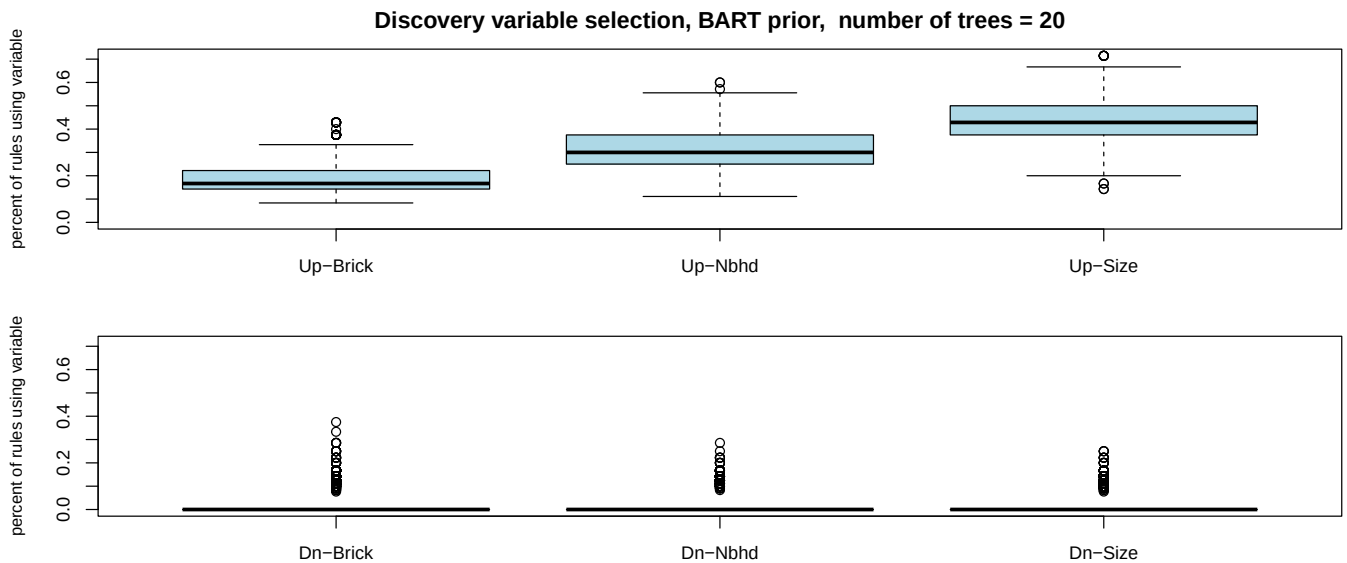
By simultaneously estimating $\hat{f}_{up}$ and $\hat{f}_{down}$, we have discovered monotonicity without any imposed assumptions!!!

This can all be most conveniently done using the mBART variable importance strategy to gauge the relationships between

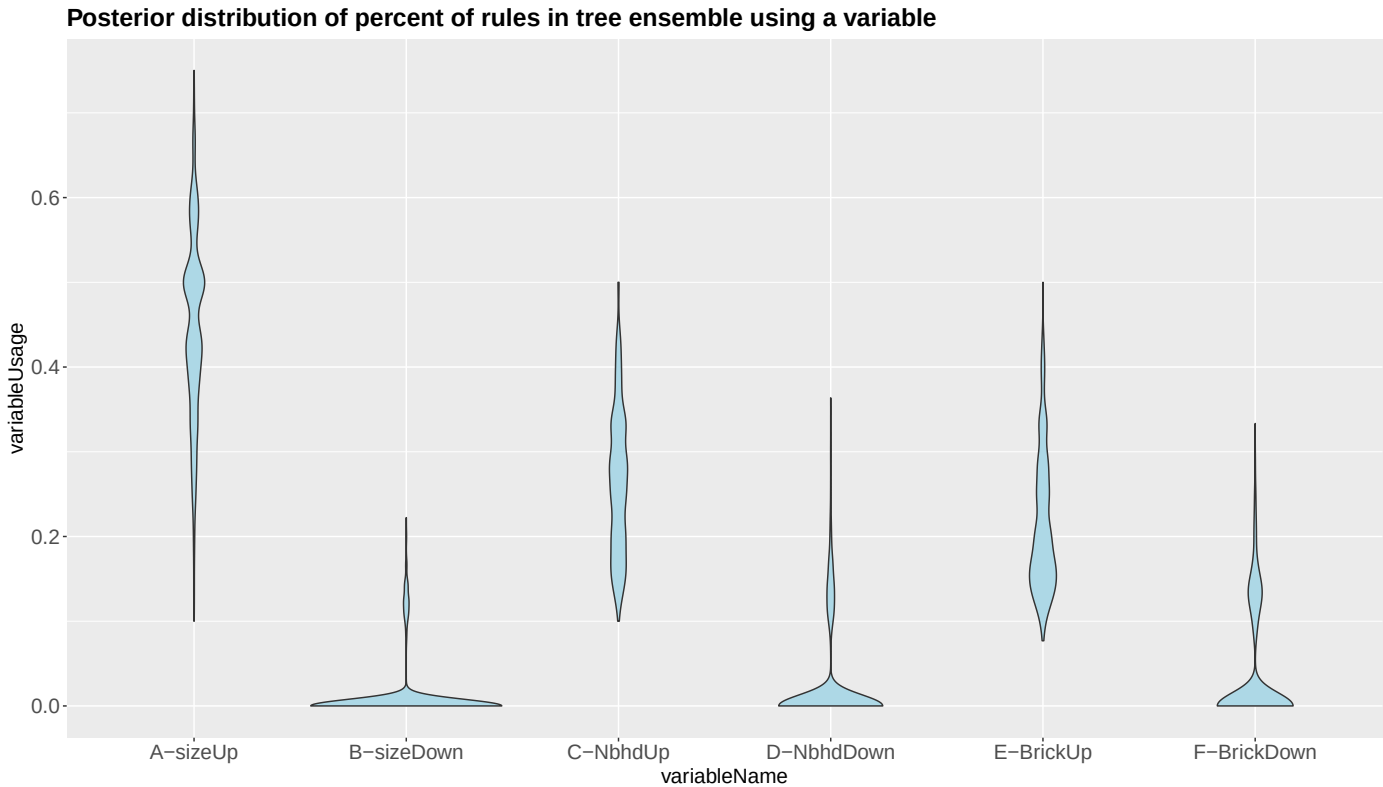
$y = price$ and

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

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!

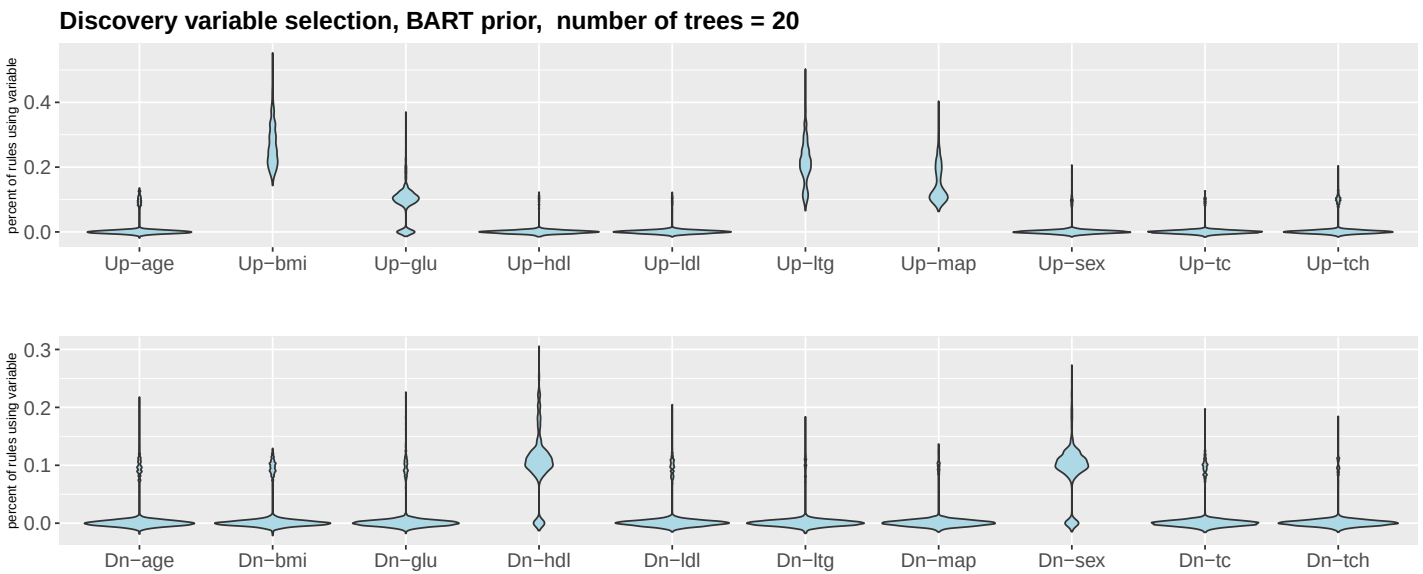

Posterior distribution of percent of rules in tree ensemble using a variable
variableUsage
0.6
0.4
0.2
0.0
A−sizeUp
B−sizeDown
C−NbhdUp
D−NbhdDown
E−BrickUp
F−BrickDown
variableName

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.

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 !!!!, Missed glu (Up) and hdl (down)

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

Hockey Penalties Example

We'll use the hockey penalty data.

The response is 1 if the current penalty is *not* on the same team as the previous penalty and 0 otherwise.

The response is called revcall or oppcall.

x is a bunch of stuff about the game situation
(the score ...).

The x values refer to the team that had the previous penalty.
For example, `goaldiff`=1 means the team that had the previous penalty is ahead by one goal.

Our response is binary and some of our predictors are categorical as well.

$P(oppcall = 1) = .6$.

Some of the important x variables:

- **inrow2**: 1 of the last two penalties where on the same team.
- **goalidff**: the lead of the last penalized team.
- **timespan**: how long since the last penalty
- **laghome**: 1 if the last penalty was on the home team.

$n \approx 60,000$.

There is a lot of fit!!!

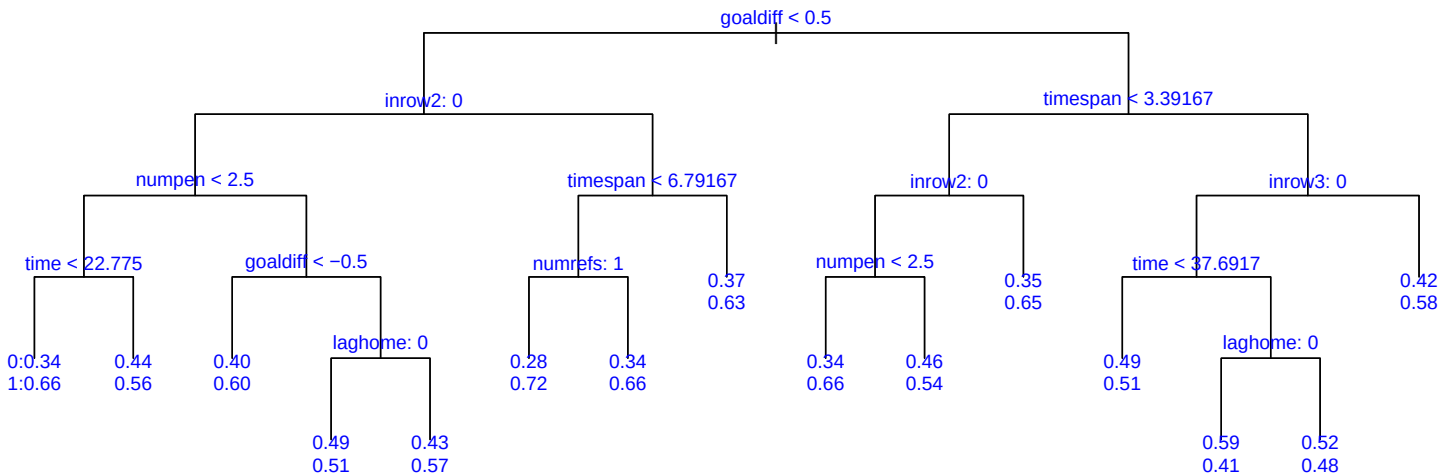
Suppose "you" got the last penalty.

if:

- if you are not winning
- you had the last two penalties
- it has not been long since the last call
- and there is only 1 referee

then:

there is a 72% chance the next call will be on the other team.



Whilst there is another game situation where the chance the next call is on the other team is only 41%.

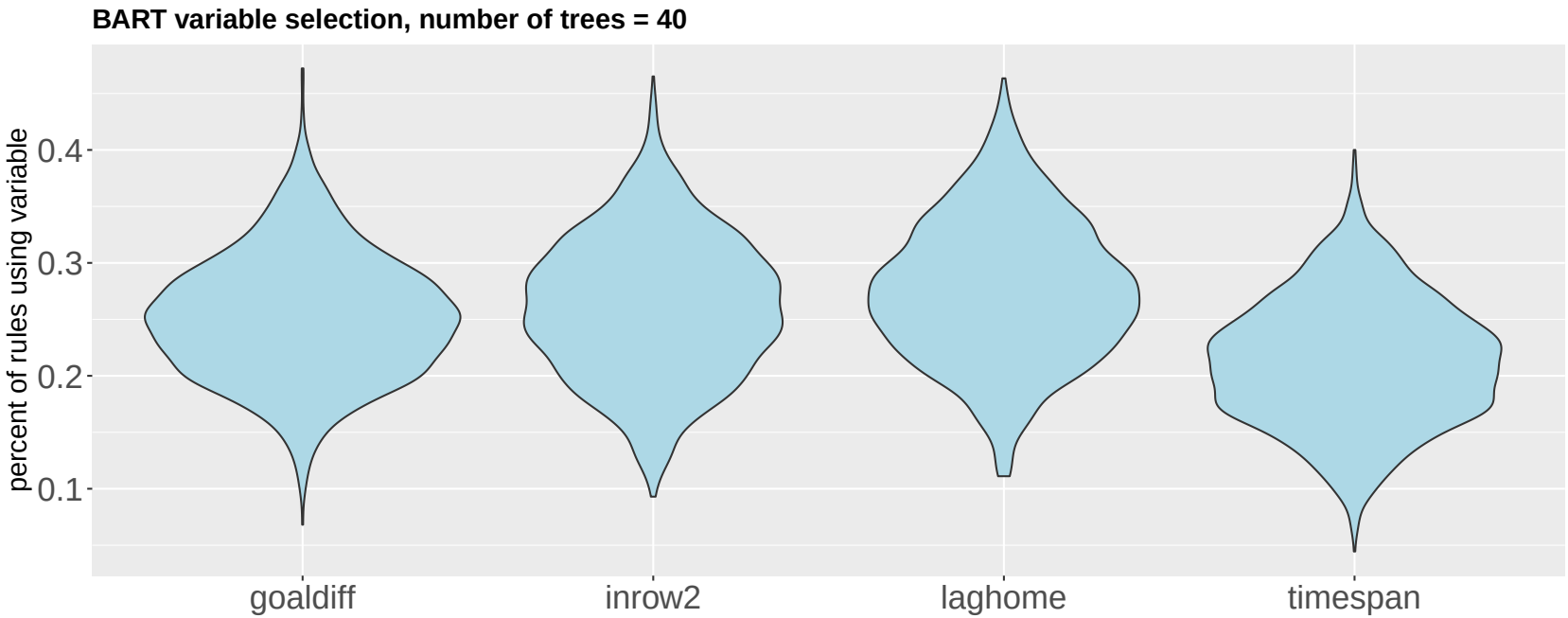
We'll use a simple subset of the data:

```
> dim(dhy)
[1] 5000    5
> head(dhy)
  oppcall  timespan laghome goaldiff inrow2
1       0 14.750000       0       -1      0
2       0  6.900000       1        2      0
3       1  8.450000       1        2      1
4       0 11.750000       0        0      0
5       1  6.300000       1        1      0
6       1  3.333333       1       -1      1
```

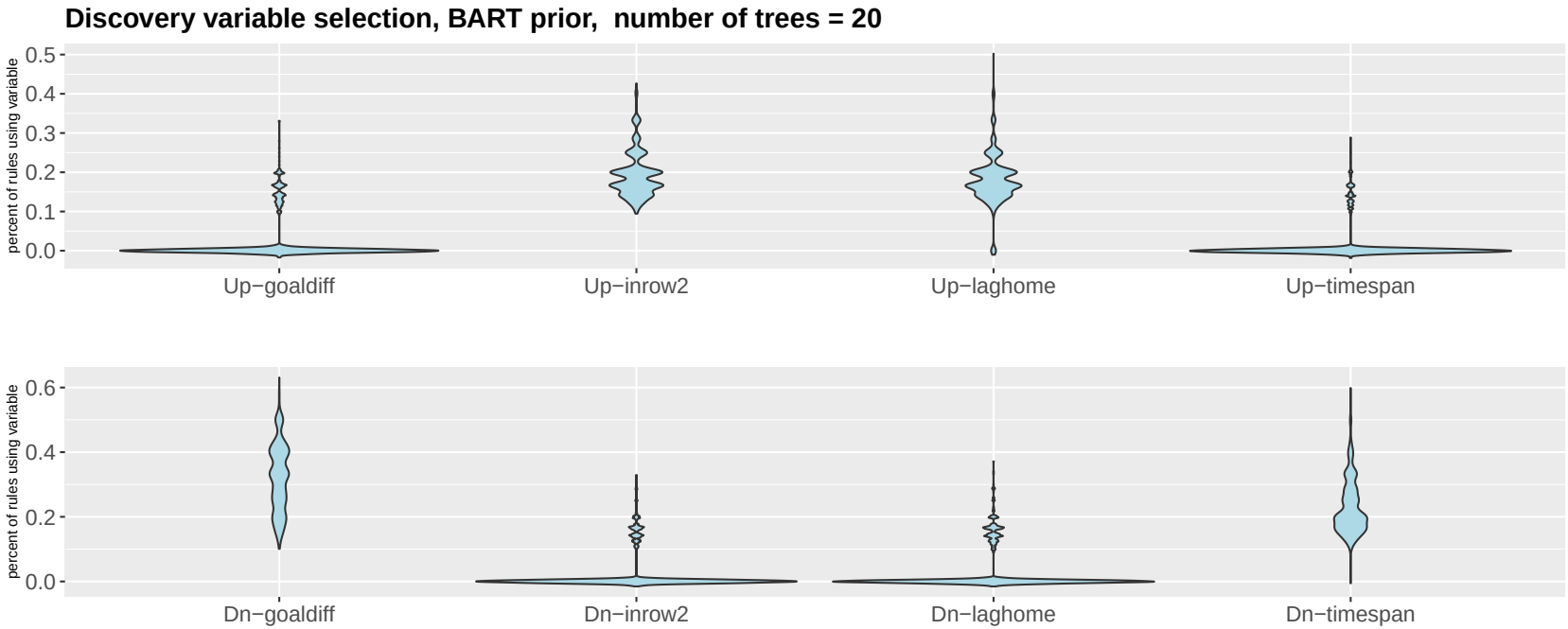
```
> mean(dhy$oppcall)
[1] 0.5904
```

We ran wbart with 40 trees.
wbart works well (in prediction) since the probability of an oppcall never gets close to 0 or 1.

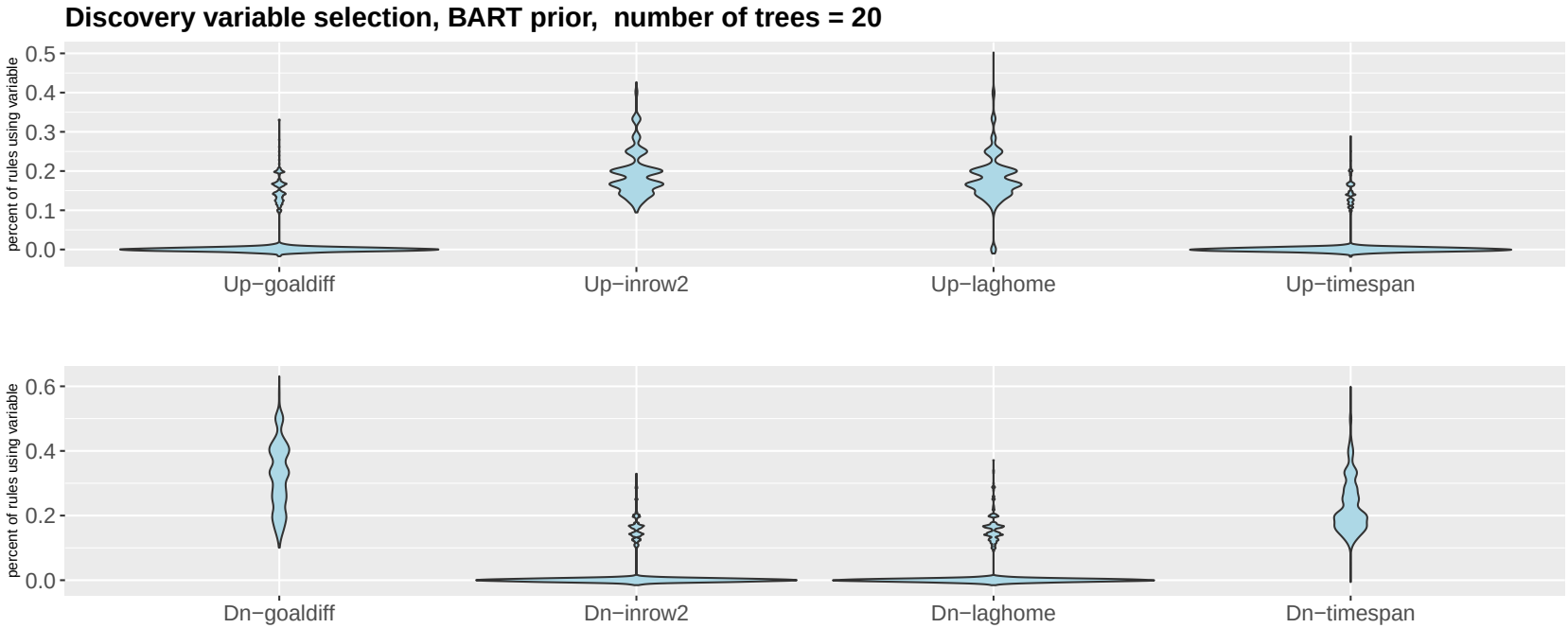
All 4 variables seem useful.



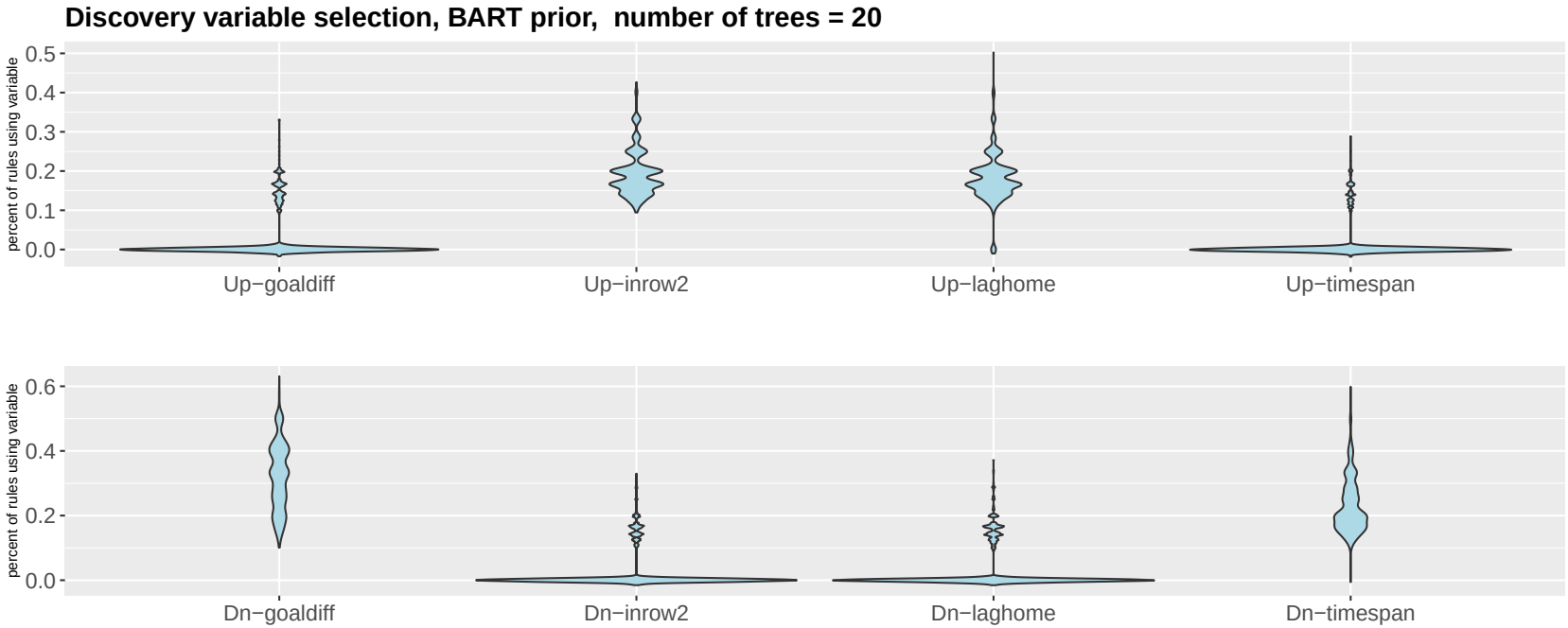
Ok, now let's do the *discovery variable selection !!*



- Only used 20 trees in the BART ensemble.
- **BART Prior**:
  Put a strong prior that the σ value is very close to the posterior mean from a default wbart run.
- **Discovery**: each of the 4 variables comes in as x and $-x$ giving 8 variable, all 8 constrained to be monotonic.



- **goaldiff**:
  If the last penalized team is ahead they are more like to be penalized again => oppcall = 0.
- **inrow2**:
  If the same team has the last two penalties it more likely that the next penalty will be on the other team => oppcall = 1.



- **laghome**:
  If the last penalized team was the home time, visitors will get the next one => oppcall = 1
- **timespan**:
  It is has been a while since the last penalty, the referee is off the hook => oppcall =0.