Using BART with a virtual species!

Getting Started

So you're interested in using `embarcadero` to do species distribution modeling with Bayesian additive regression trees! That's great. BARTs are a powerful way to do machine learning and, while not a new method per se, they are very new for SDMs.

Most of the core functionality of `embarcadero` is actually a wrapper for `dbarts`, which runs the actual BART fitting process. This vignete will show you

1. How to run BARTs
2. Variable importance measures
3. Automated variable selection
4. Partial dependence plots
5. Visualizing the posterior distribution

There's also just going to be some general comments on the process of using BARTs, the challenges to working with them, and some things that are hopefully coming next.

If you want to install, do it using devtools for now:

```
#devtools::install_github('cjcarlson/embarcadero')
library(embarcadero, quietly = T)
#>
#> Attaching package: 'raster'
#> The following object is masked from 'package:dplyr':
#>
#>     select
#> The following object is masked from 'package:tidyr':
#>
#>     extract
#>
#> Attaching package: 'dbarts'
#> The following object is masked from 'package:raster':
#>
#>     extract
#> The following object is masked from 'package:tidyr':
#>
#>     extract
#>
#> Attaching package: 'gplots'
#> The following object is masked from 'package:stats':
#>
#>     lowess
#>
#> Attaching package: 'patchwork'
#> The following object is masked from 'package:raster':
#>
#>     area
#>
#> Attaching package: 'cowplot'
#> The following object is masked from 'package:patchwork':
#>
```

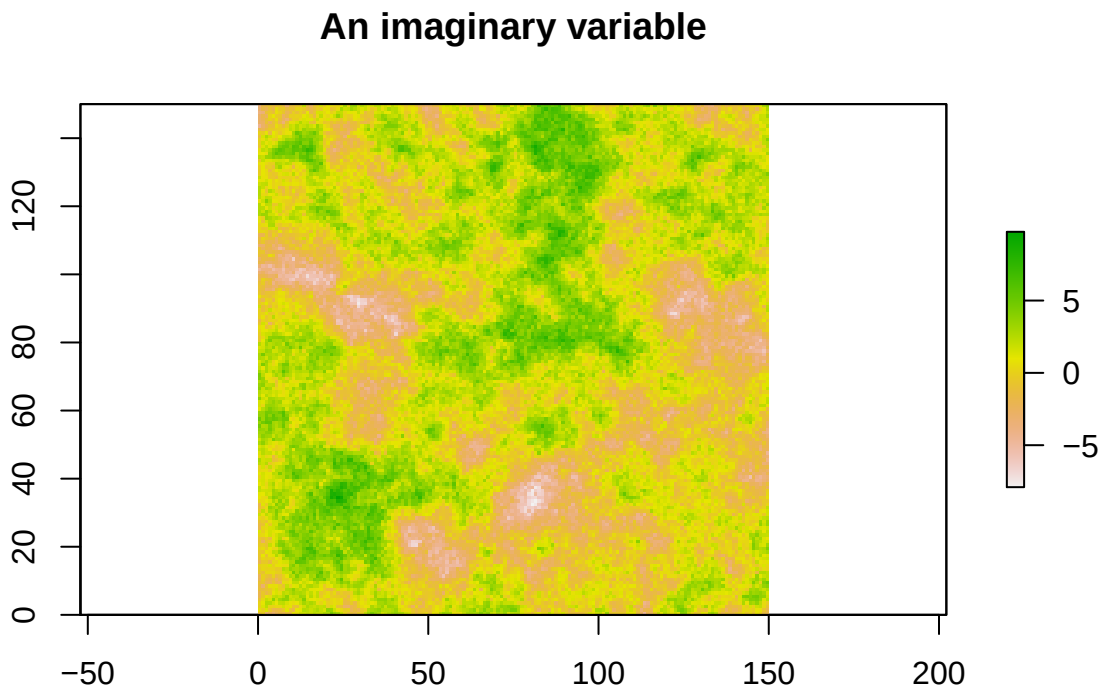
```
#>     align_plots
#> The following object is masked from 'package:ggplot2':
#>
#>     ggsave
#>
#> Attaching package: 'magrittr'
#> The following object is masked from 'package:dbarts':
#>
#>     extract
#> The following object is masked from 'package:raster':
#>
#>     extract
#> The following object is masked from 'package:purrr':
#>
#>     set_names
#> The following object is masked from 'package:tidyr':
#>
#>     extract
#>
#> Attaching package: 'ggpubr'
#> The following object is masked from 'package:cowplot':
#>
#>     get_legend
#> The following object is masked from 'package:raster':
#>
#>     rotate
#>
#> Attaching package: 'matrixStats'
#> The following object is masked from 'package:dplyr':
#>
#>     count
#>
#> Attaching package: 'data.table'
#> The following object is masked from 'package:raster':
#>
#>     shift
#> The following objects are masked from 'package:dplyr':
#>
#>     between, first, last
#> The following object is masked from 'package:purrr':
#>
#>     transpose
#>
#> Attaching package: 'embarcadero'
#> The following object is masked from 'package:purrr':
#>
#>     partial
library(dismo, quietly=T)
library(NLMR, quietly = T)
library(raster, quietly = T)
library(virtualspecies, quietly = T)
set.seed(42)
```

Doors are closing; please stand clear of the doors.

Creating the virtual species

First, let's create an imaginary landscape. We do this using the NLMR package:

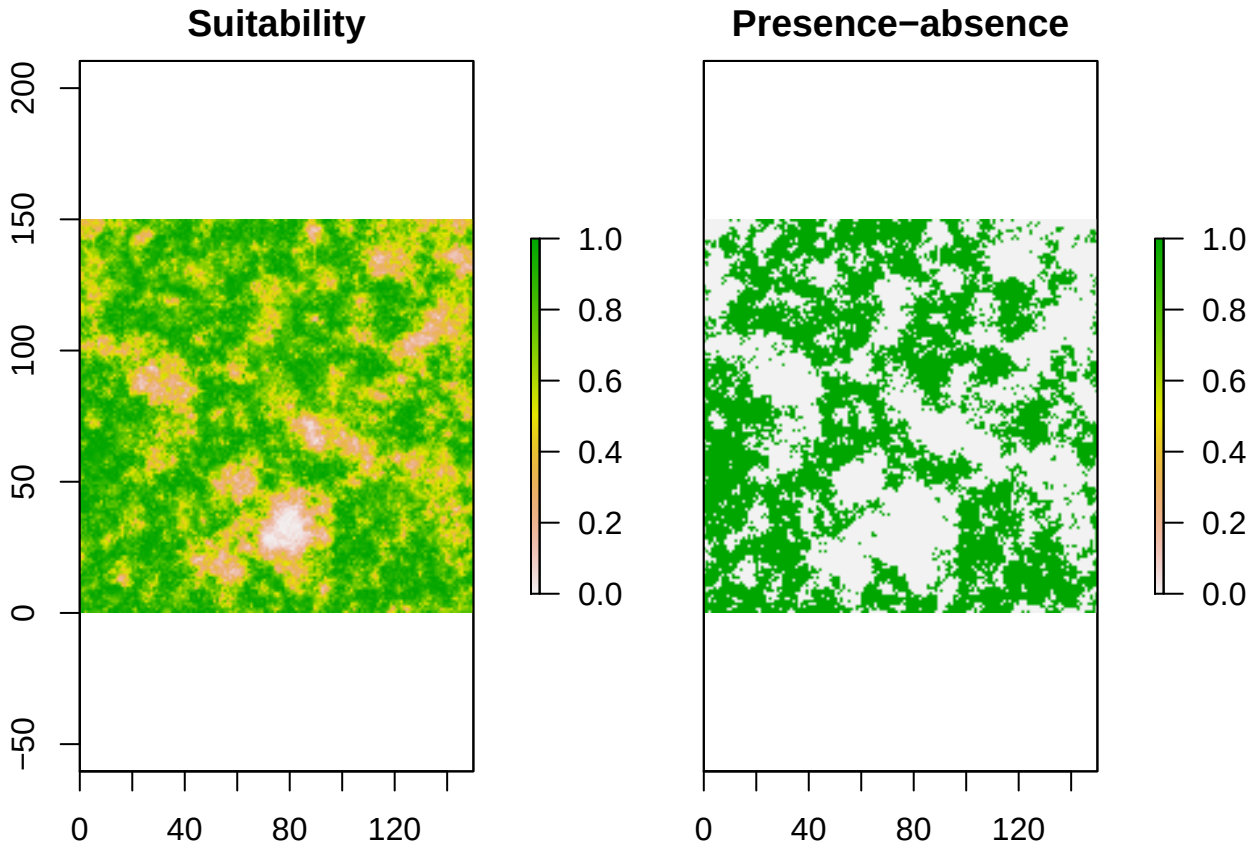
```
onelandscape <- function(x) {NLMR::nlm_gaussianfield(nrow = 150,
                                                     ncol = 150,
                                                     rescale = FALSE)}
climate <- stack(lapply(c(1:8), onelandscape))
xnames <- c('x1','x2','x3','x4','x5','x6','x7','x8')
names(climate) <- xnames

plot(climate[[1]],main='An imaginary variable')
```



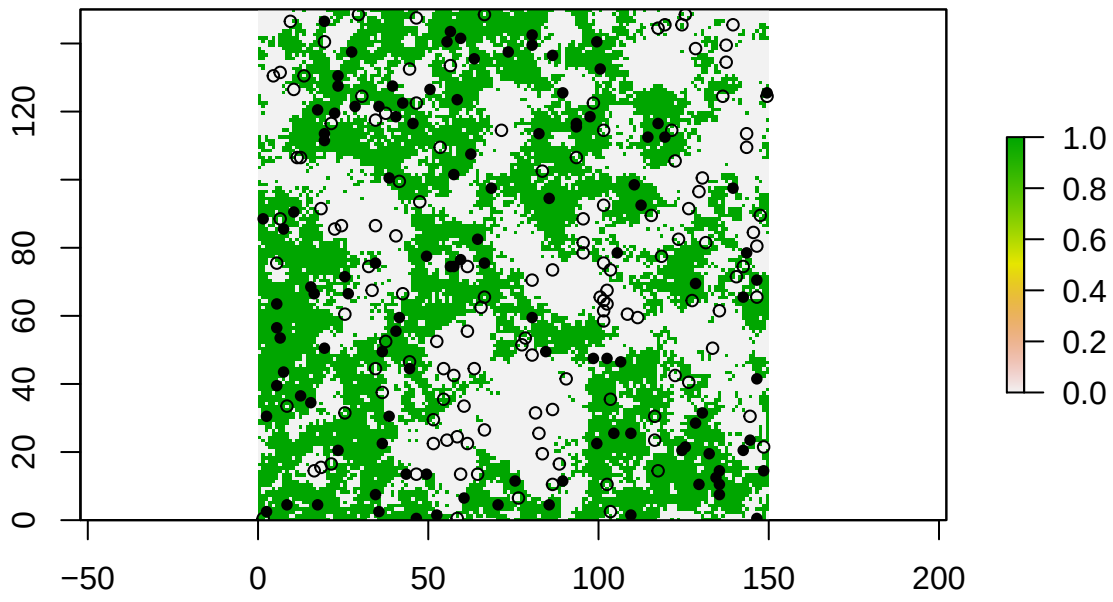
Next, let's make a species using the 'virtualspecies' package. Our imaginary species will only responds to variables 1-4, making variables 5-8 uninformative predictors (hopefully our model will drop them):

```
# Generate the species' climatic niche

random.sp <- generateRandomSp(climate[[1:4]],
                              # ^ These are the informative predictors
                              approach="pca",
                              relations='gaussian',
                              species.prevalence=0.5,
                              realistic.sp = TRUE,
                              PA.method='threshold')
#>   - Perfoming the pca
#>   - Defining the response of the species along PCA axes
```

```
#>  - Calculating suitability values
#>    The final environmental suitability was rescaled between 0 and 1.
#>                    To disable, set argument rescale = FALSE
#>  - Converting into Presence - Absence
#>    Threshold conversion finished:
#>
#> - cutoff = 0.75970665861952
#> - species prevalence =0.5
```




```
# Generate some presences, and some absences, with imperfect detection

sp.points <- sampleOccurrences(random.sp,
                               n=250,
                               type = 'presence-absence',
                               detection.probability = 0.9)
```




```
# Extract the associated climate values

occ <- SpatialPoints(sp.points$sample.points[,c('x','y')])
occ.df <- cbind(sp.points$sample.points,
                raster::extract(climate, occ))

# Finally, let's drop the long-lats and the "Real" ground truthed presence-absence values, and just lea

occ.df <- occ.df[,-c(1:3)]
```

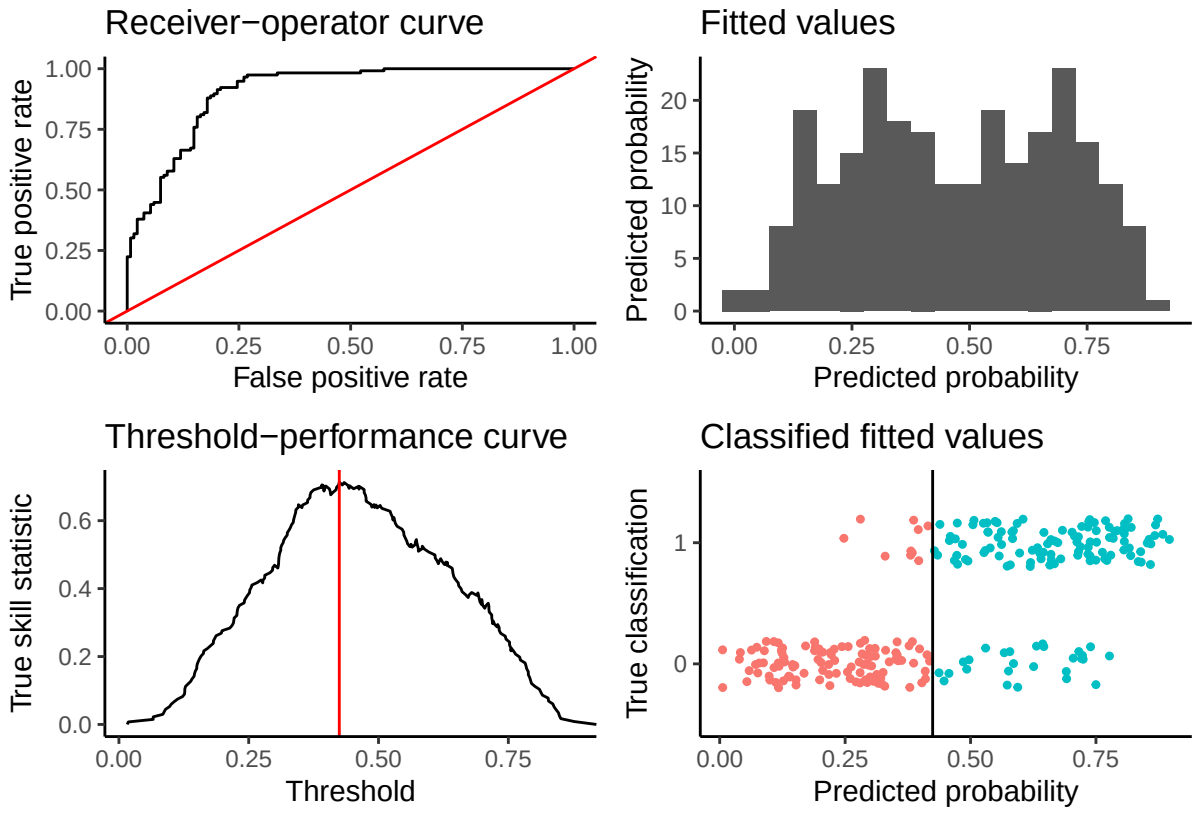
Alright. Now that we have the dataset, let's get to modeling!

We could easily throw all our data in one model, run it on defaults, make a map, and never think about it again. There's no laws against it.

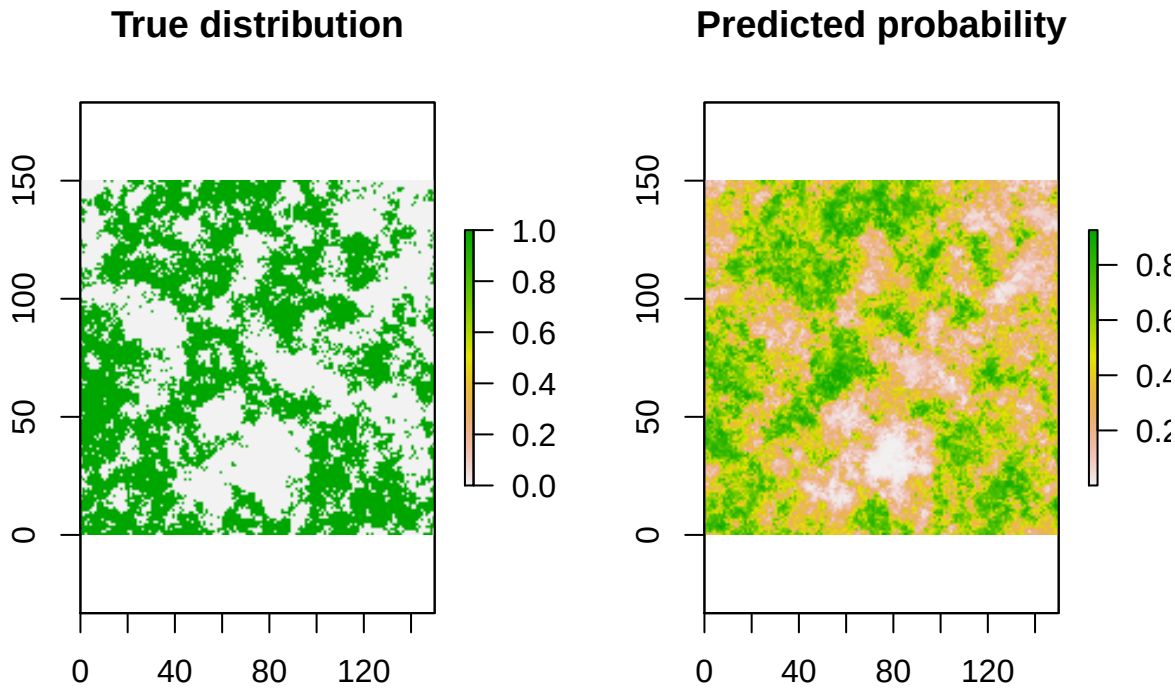
```
# Check out the data structure
head(occ.df)
#>   Observed  x1     x2     x3    x4    x5    x6    x7    x8
#> 1        0 1.9  0.093 -3.935  0.45 -1.90  0.16  4.97 -1.23
#> 2        1 1.4 -1.396  1.825 -1.43  2.27 -1.48  1.19  3.96
#> 3        0 3.9 -1.202 -0.964  2.15 -2.24  5.85  1.46  5.12
#> 4        0 1.7 -1.624 -2.984  2.75  3.08  3.84 -1.93  0.97
#> 5        1 2.5  1.362  0.089 -4.69 -0.96  0.28  0.66  2.61
#> 6        0 1.4  3.856 -1.720  0.70 -0.54 -2.50 -0.92  6.05

# Train the model
sdm <- bart(y.train=occ.df[,'Observed'],
```

```
            x.train=occ.df[,xnames],
            keeptrees = TRUE) # It's very important this is set to TRUE
#>
#> Running BART with binary y
#>
#> number of trees: 200
#> number of chains: 1, number of threads 1
#> Prior:
#>  k: 2.000000
#>  power and base for tree prior: 2.000000 0.950000
#>  use quantiles for rule cut points: false
#> data:
#>  number of training observations: 250
#>  number of test observations: 0
#>  number of explanatory variables: 8
#>
#> Cutoff rules c in x<=c vs x>c
#> Number of cutoffs: (var: number of possible c):
#> (1: 100) (2: 100) (3: 100) (4: 100) (5: 100)
#> (6: 100) (7: 100) (8: 100)
#>
#> offsets:
#>  reg : 0.00 0.00 0.00 0.00 0.00
#> Running mcmc loop:
#> iteration: 100 (of 1000)
#> iteration: 200 (of 1000)
#> iteration: 300 (of 1000)
#> iteration: 400 (of 1000)
#> iteration: 500 (of 1000)
#> iteration: 600 (of 1000)
#> iteration: 700 (of 1000)
#> iteration: 800 (of 1000)
#> iteration: 900 (of 1000)
#> iteration: 1000 (of 1000)
#> total seconds in loop: 2.056507
#>
#> Tree sizes, last iteration:
#> [1] 2 2 2 2 2 2 2 2 2 2 2 3 2 3 2 3 2 3
#> 2 2 3 2 3 3 4 4 2 2 2 2 3 2 2 2 2 4 2 2
#> 2 3 3 2 2 1 2 2 2 3 2 4 3 2 3 3 2 2 1 2
#> 2 2 2 2 2 4 2 2 3 3 3 1 2 2 5 1 4 3 2 2
#> 2 2 3 2 2 3 4 2 3 2 2 3 2 2 3 2 2 3 2 2
#> 2 3 4 2 3 4 4 2 3 2 2 3 2 3 2 2 2 2 2 3
#> 2 2 2 3 2 1 2 2 3 2 1 3 2 2 4 2 2 3 2 1
#> 2 2 2 3 3 3 3 1 1 2 1 3 3 1 2 2 3 3 2 2
#> 3 3 3 4 3 4 1 3 2 2 2 2 2 3 2 2 3 4 2 2
#> 3 2 2 2 2 3 2 4 2 2 3 3 2 2 2 3 2 3 2 3
#> 3 2
#>
#> Variable Usage, last iteration (var:count):
#> (1: 31) (2: 34) (3: 29) (4: 32) (5: 45)
#> (6: 37) (7: 36) (8: 34)
#> DONE BART
```

```
# Model diagnostics
summary(sdm)
#> Call:  bart occ.df[, xnames] occ.df[, "Observed"] TRUE
#>
#> Predictor list:
#>  x1 x2 x3 x4 x5 x6 x7 x8
#>
#> Area under the receiver-operator curve
#> AUC = 0.91
#>
#> Recommended threshold (maximizes true skill statistic)
#> Cutoff =  0.42
#> TSS =  0.71
#> Resulting type I error rate:  0.078
#> Resulting type II error rate:  0.21
```

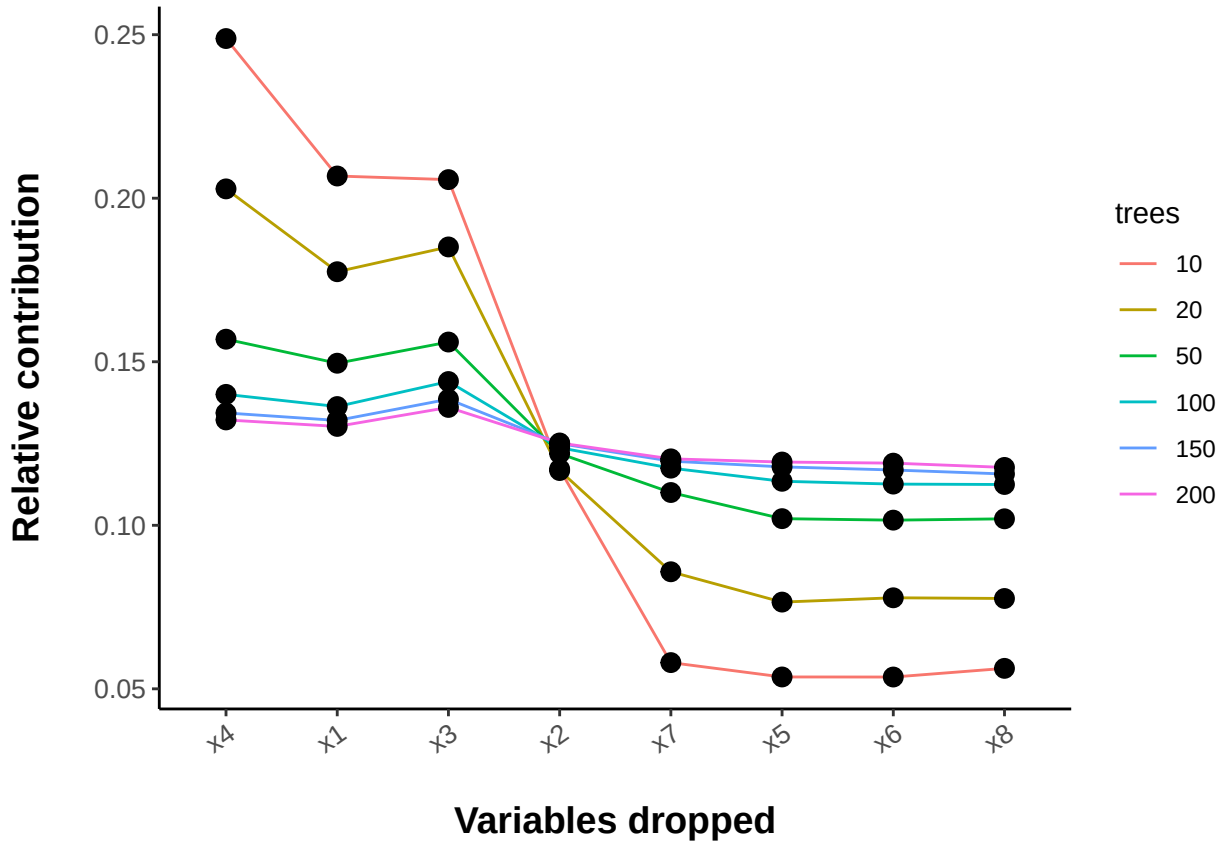



```
# Predict the species distribution!
map <- predict(sdm, climate, quiet=TRUE)

# How's it look?
par(mfrow=c(1,2))
plot(random.sp$pa.raster, main='True distribution')
plot(map, main='Predicted probability')
```



This is.... alright? We could probably do better. One of the easiest ways is to cut some variables we don't think are performing well. In BART, pushing the models towards a smaller number of trees forces the variables to compete a bit, and preferentially upweights the better ones. We can plot that!

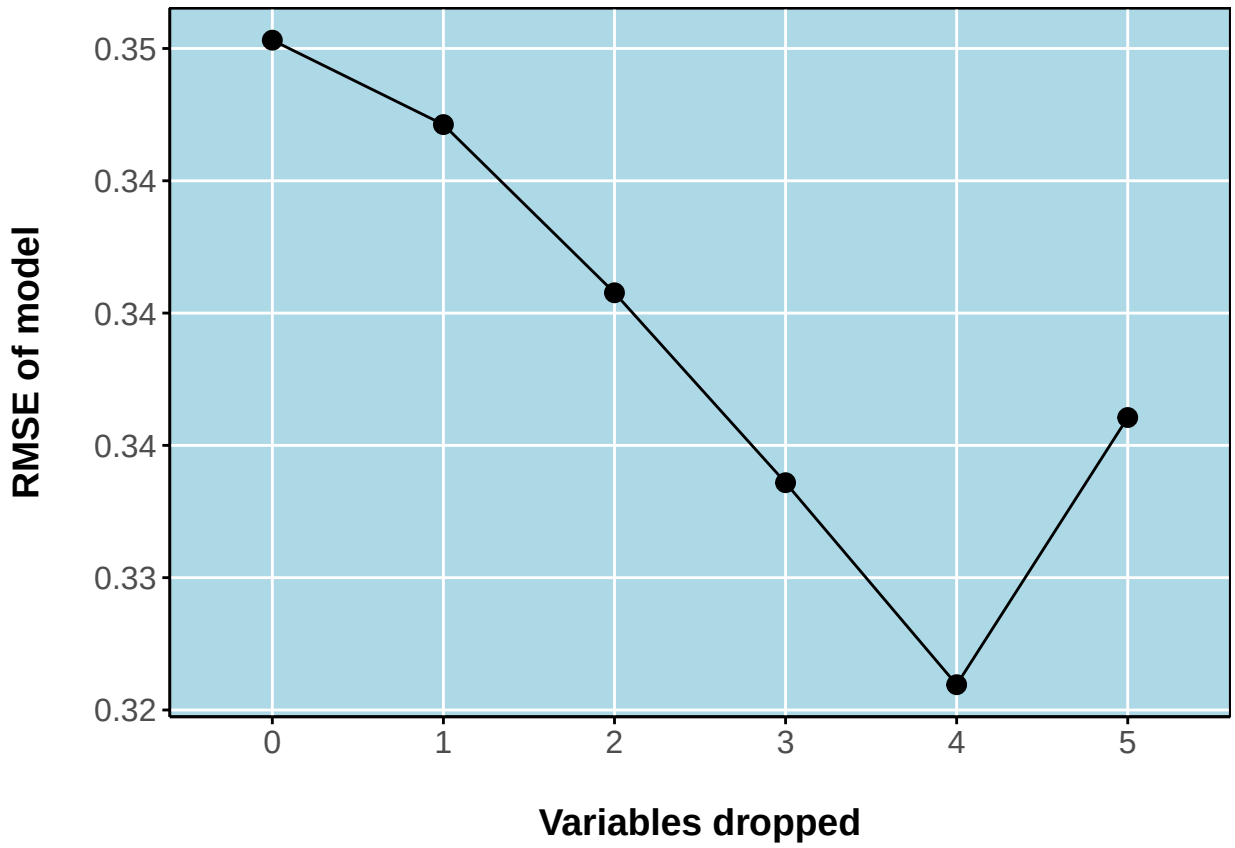
```
# A variable importance diagnostic. What's behaving well?
# This takes a while to run normally! Drop the iter if you want a plot faster with more variance.
varimp.diag(occ.df[,xnames], occ.df['Observed'], iter=50, quiet=TRUE)
#>
#>  10 tree models: 50 iterations
#>
#>  20 tree models: 50 iterations
#>
#>  50 tree models: 50 iterations
#>
#>  100 tree models: 50 iterations
#>
#>  150 tree models: 50 iterations
#>
#>  200 tree models: 50 iterations
```



OK. This would suggest that some of the variables, like x5 and x7, are less important while x3 and x4 are more important. But that doesn't give us a concrete list of what to drop. Luckily, embarcadero has a stepwise variable set reduction function. Let's run that, and then retrain the model.

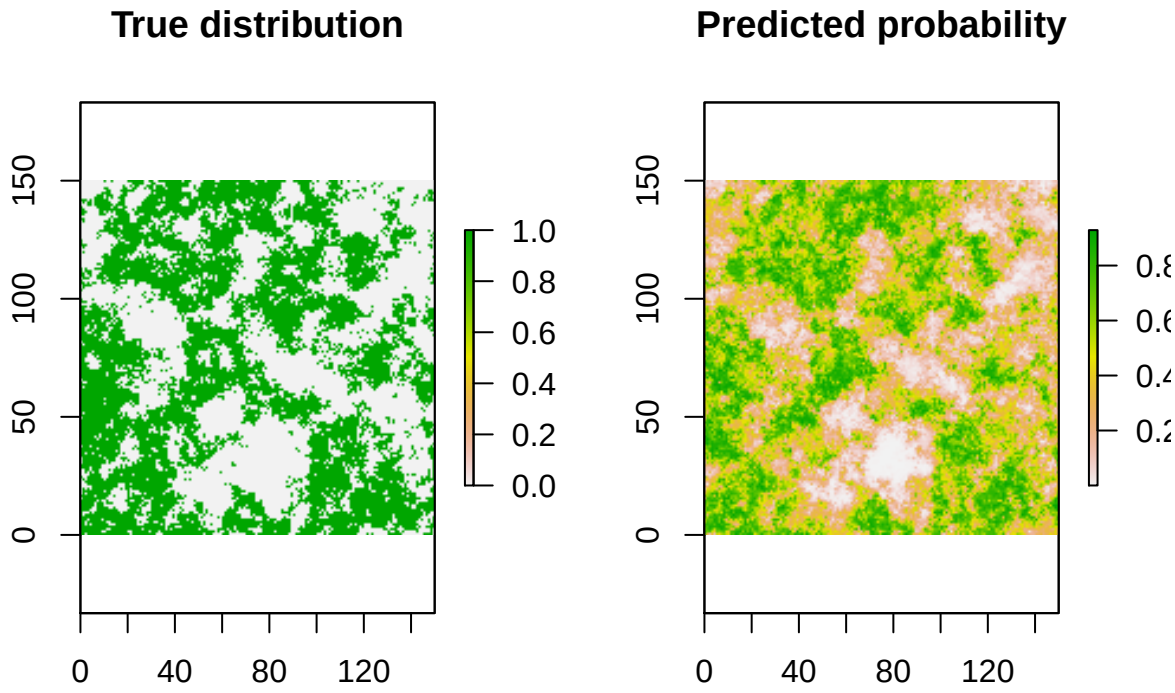
```
# Stepwise variable set reduction
step.model <- variable.step(x.data=occ.df[,xnames],
                            y.data=occ.df['Observed'],
                            quiet=TRUE)
#> [1] Number of variables included: 8
#> [1] Dropped:
#> [1]
#> [1] ---------------------------------------
#> [1] Number of variables included: 7
#> [1] Dropped:
#> [1] x6
#> [1] ---------------------------------------
#> [1] Number of variables included: 6
#> [1] Dropped:
#> [1] x6 x5
#> [1] ---------------------------------------
#> [1] Number of variables included: 5
#> [1] Dropped:
#> [1] x6 x5 x8
#> [1] ---------------------------------------
#> [1] Number of variables included: 4
#> [1] Dropped:
#> [1] x6 x5 x8 x7
```

```
#> [1] ---------------------------------------
#> [1] Number of variables included: 3
#> [1] Dropped:
#> [1] x6 x5 x8 x7 x2
#> [1] ---------------------------------------
```



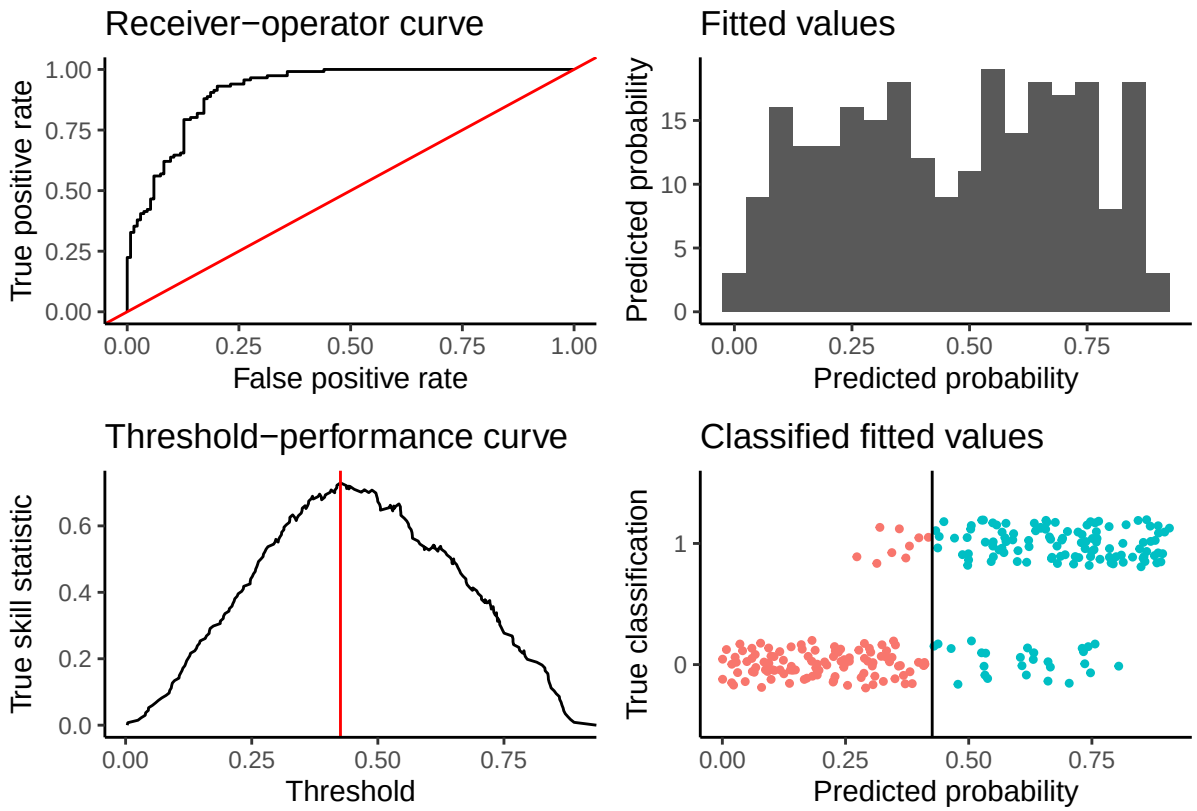

```
#> [1] ----------------------------------------
#> [1] Final recommended variable list
#> [1] x1 x2 x3 x4
step.model
#> [1] "x1" "x2" "x3" "x4"

# Retrain the model
sdm <- bart(x.train=occ.df[, step.model], y.train=occ.df[,'Observed'],
            keeptrees = TRUE)
#>
#> Running BART with binary y
#>
#> number of trees: 200
#> number of chains: 1, number of threads 1
#> Prior:
#>  k: 2.000000
#>  power and base for tree prior: 2.000000 0.950000
#>  use quantiles for rule cut points: false
#> data:
#>  number of training observations: 250
```

```
#>  number of test observations: 0
#>  number of explanatory variables: 4
#>
#> Cutoff rules c in x<=c vs x>c
#> Number of cutoffs: (var: number of possible c):
#> (1: 100) (2: 100) (3: 100) (4: 100)
#>
#> offsets:
#>  reg : 0.00 0.00 0.00 0.00 0.00
#> Running mcmc loop:
#> iteration: 100 (of 1000)
#> iteration: 200 (of 1000)
#> iteration: 300 (of 1000)
#> iteration: 400 (of 1000)
#> iteration: 500 (of 1000)
#> iteration: 600 (of 1000)
#> iteration: 700 (of 1000)
#> iteration: 800 (of 1000)
#> iteration: 900 (of 1000)
#> iteration: 1000 (of 1000)
#> total seconds in loop: 2.181768
#>
#> Tree sizes, last iteration:
#> [1] 4 3 2 2 4 2 3 2 2 2 3 3 2 3 2 2 3 2
#> 2 3 2 4 2 2 3 3 2 2 2 3 2 2 4 3 3 3 2 2
#> 2 2 2 3 2 3 2 3 2 2 2 5 2 5 1 3 3 3 3 3
#> 6 2 1 2 2 2 2 2 3 3 3 2 2 2 2 2 2 3 2 3
#> 5 3 2 2 2 1 2 3 2 3 2 2 2 2 2 3 2 4 4 2
#> 2 2 1 2 3 2 3 1 2 1 3 2 3 2 2 5 3 3 2 2
#> 2 2 3 2 4 4 2 3 3 2 5 5 3 3 3 3 2 3 2 2
#> 2 2 2 2 2 2 2 2 2 2 2 4 2 2 3 2 3 4 3 2
#> 3 2 5 2 3 2 3 3 3 2 2 2 2 2 2 2 3 3 2 2
#> 2 3 2 2 2 2 2 2 2 2 2 3 2 3 2 2 3 2 3 3
#> 2 2
#>
#> Variable Usage, last iteration (var:count):
#> (1: 74) (2: 75) (3: 67) (4: 83)
#> DONE BART

# Predict the species distribution!
map <- predict(sdm, climate, quiet=TRUE)

# How's it look?
par(mfrow=c(1,2))
plot(random.sp$pa.raster, main='True distribution')
plot(map, main='Predicted probability')
```

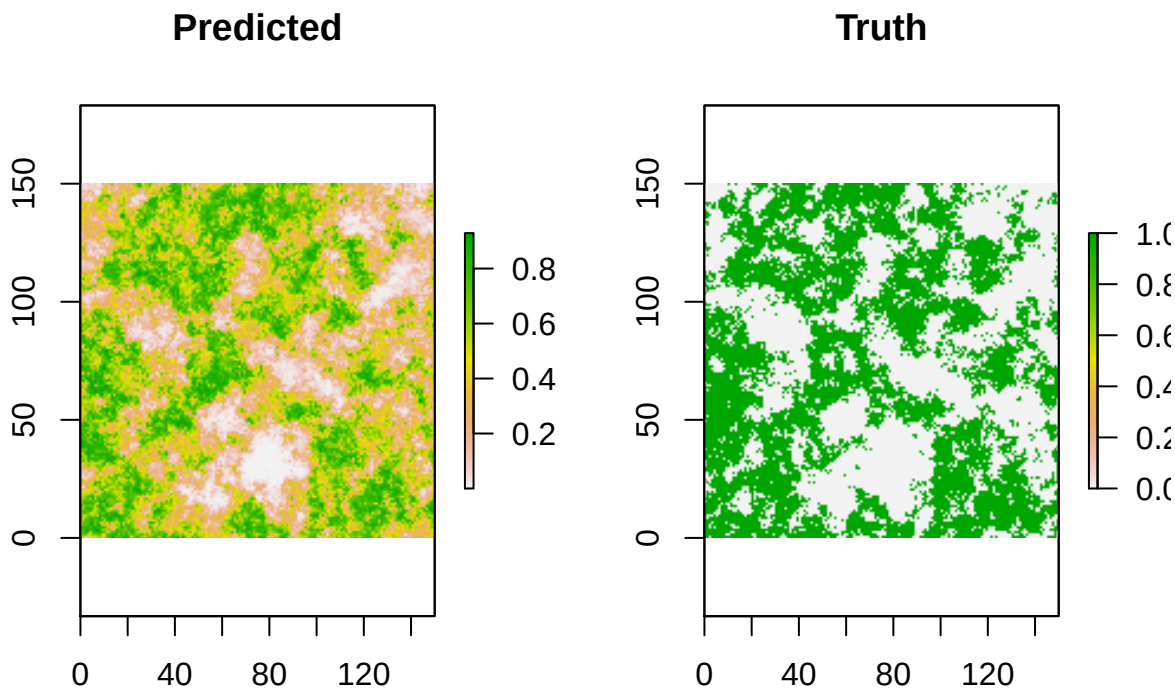


OK. We have a good-looking model. What's inside?

```
# How good is it?
summary(sdm)
#> Call:  bart occ.df[, step.model] occ.df[, "Observed"] TRUE
#>
#> Predictor list:
#>  x1 x2 x3 x4
#>
#> Area under the receiver-operator curve
#> AUC = 0.92
#>
#> Recommended threshold (maximizes true skill statistic)
#> Cutoff =  0.43
#> TSS =  0.73
#> Resulting type I error rate:  0.069
#> Resulting type II error rate:  0.2
```



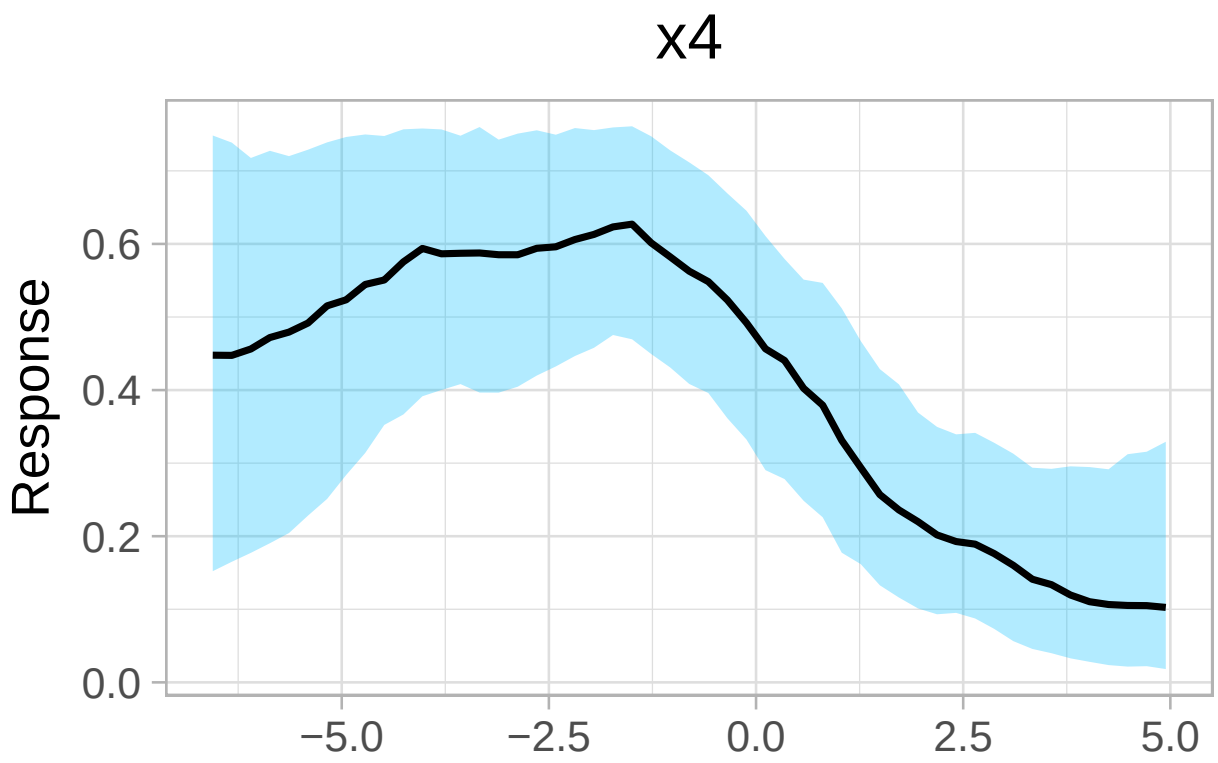

```
map <- predict(sdm, climate, quantiles=c(0.025, 0.975), quiet=TRUE)

# Check
par(mfrow=c(1,2))
plot(map[[1]], main='Predicted')
plot(random.sp$pa.raster, main='Truth')
```



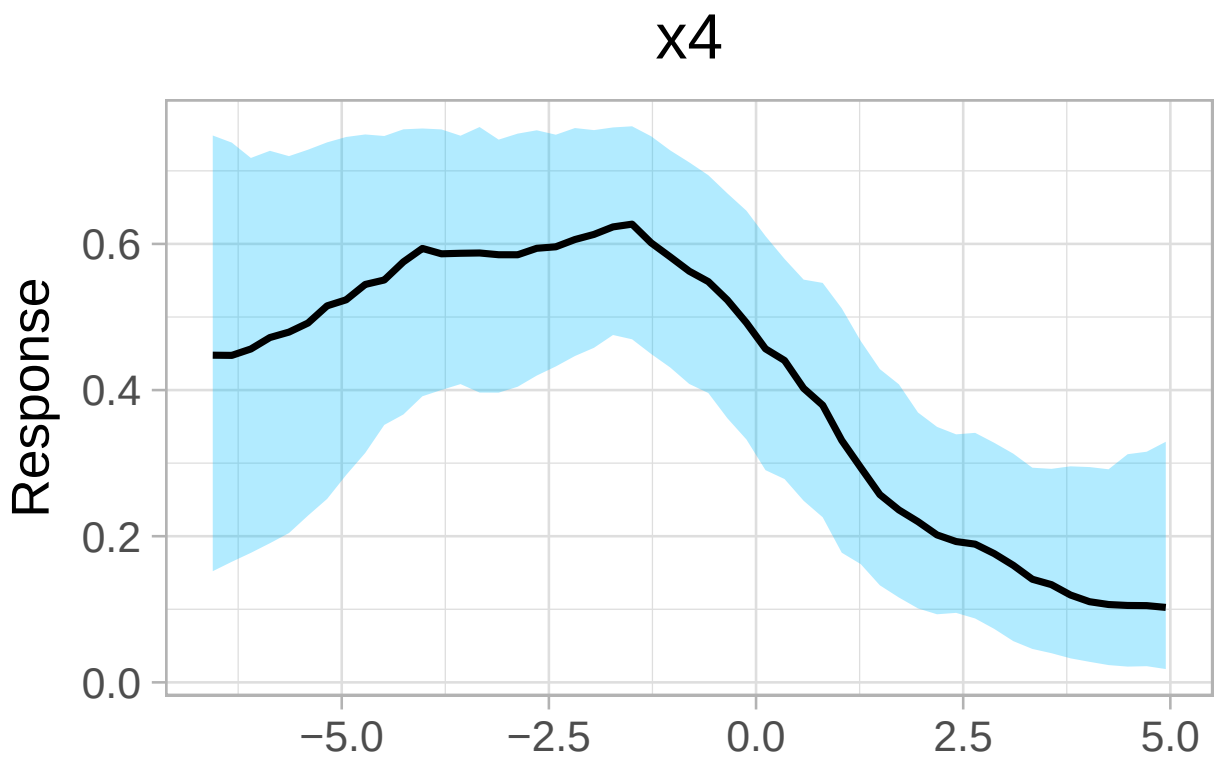

```
#plot(values(map) ~ values(random.sp$pa.raster))
```

OK. This part involves comparing partials to gbm.

```
partial(sdm, x.vars=c('x4'),
        smooth=5,
        equal=TRUE,
        trace=FALSE)
```



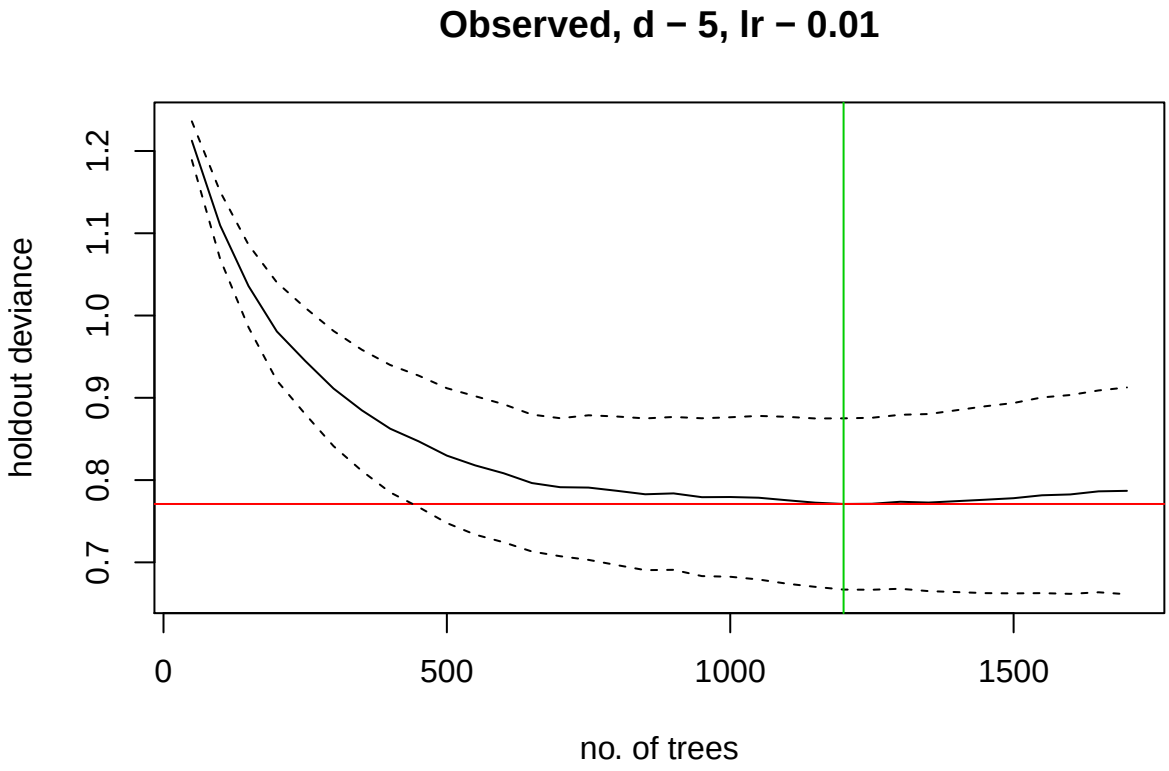

```
#> [[1]]
```



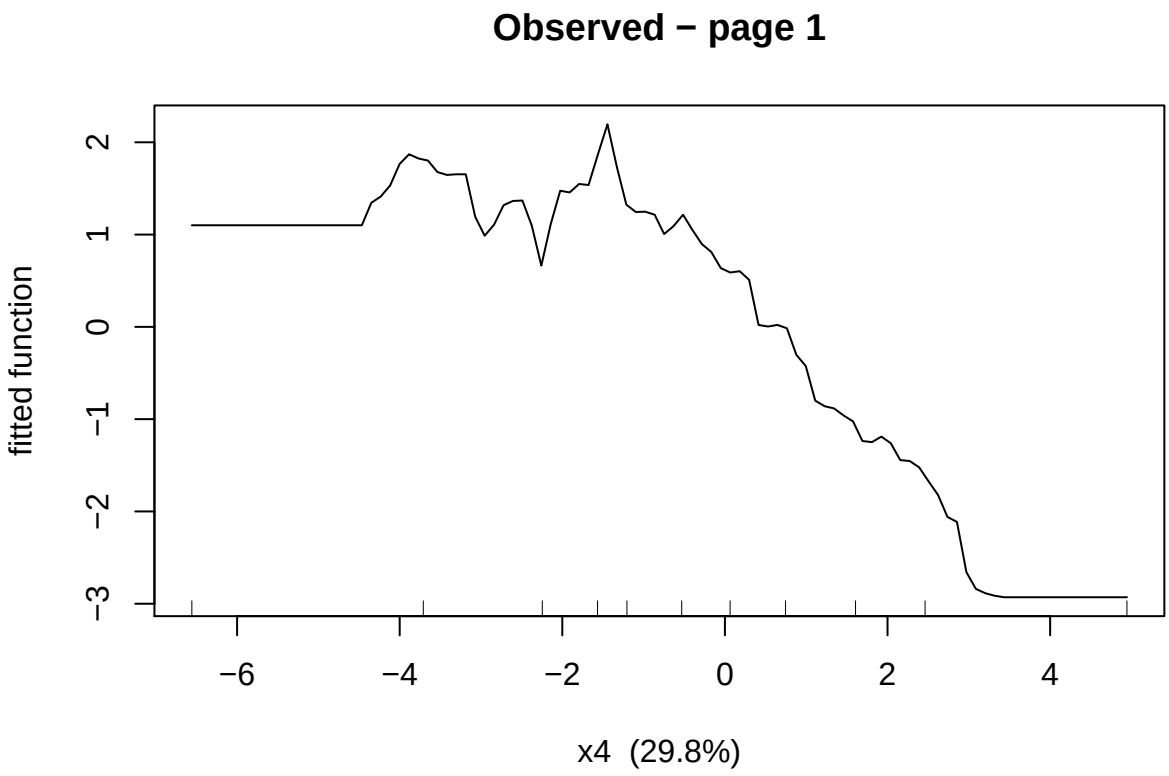
gbm time

```
gbm1 <- gbm.step(data=occ.df, gbm.x = 2:5, gbm.y = 1,
                 family = "bernoulli",
                 tree.complexity = 5,
                 learning.rate = 0.01,
                 bag.fraction = 0.5)
#> Loading required namespace: gbm
#>
#>
#>  GBM STEP - version 2.9
#>
#> Performing cross-validation optimisation of a boosted regression tree model
#> for Observed and using a family of bernoulli
#> Using 250 observations and 4 predictors
#> creating 10 initial models of 50 trees
#>
#>  folds are stratified by prevalence
#> total mean deviance =  1.4
#> tolerance is fixed at  0.0014
#> ntrees resid. dev.
#> 50    1.2
#> now adding trees...
#> 100   1.1
#> 150   1
#> 200   0.98
#> 250   0.94
```

```
#> 300   0.91
#> 350   0.88
#> 400   0.86
#> 450   0.85
#> 500   0.83
#> 550   0.82
#> 600   0.81
#> 650   0.8
#> 700   0.79
#> 750   0.79
#> 800   0.79
#> 850   0.78
#> 900   0.78
#> 950   0.78
#> 1000   0.78
#> 1050   0.78
#> 1100   0.78
#> 1150   0.77
#> 1200   0.77
#> 1250   0.77
#> 1300   0.77
#> 1350   0.77
#> 1400   0.77
#> 1450   0.78
#> 1500   0.78
#> 1550   0.78
#> 1600   0.78
#> 1650   0.79
#> 1700   0.79
#> Warning: glm.fit: algorithm did not converge
#> Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
#> Warning: glm.fit: algorithm did not converge
#> Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
#> fitting final gbm model with a fixed number of 1200 trees for Observed
#> Warning: glm.fit: algorithm did not converge

#> Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```




```
#>
#> mean total deviance = 1.4
#> mean residual deviance = 0.2
#>
#> estimated cv deviance = 0.77 ; se = 0.1
#>
#> training data correlation = 0.97
#> cv correlation =  0.72 ; se = 0.047
#>
#> training data AUC score = 1
#> cv AUC score = 0.90 ; se = 0.026
#>
#> elapsed time -  0.11 minutes
par(mfrow=c(1,1))
gbm.plot(gbm1, variable.no=4, rug=TRUE,
         #main="BRT partial",
         plot.layout=c(1,1))
```


Observed − page 1
fitted function
2
1
0
−1
−2
−3
−6
−4
−2
0
2
4
x4 (29.8%)