LSIRM Statistical/Machine Learning
Tree-based Regression Models

Dave Armstrong

University of Western Ontario
Department of Political Science

e: dave.armstrong@uwo.ca
w: www.quantoid.net/teachwlu/

Classification and Regression Trees (CART)

CART works in a decision-tree framework.

- Considering all independent variables, which dichotomization on one of them explains the most variance.
- Conditional on the previous *split*, which next dichotomization explains the most variance.
- Loss function is the well-known residual sum of squares.
- Continue until some stopping rule is met.

Notation

$$f(X_i) = T(X_i, \Theta) \equiv \sum_{b=1}^{B} c_b I(X_i \in R_b)$$

- $T()$ is a regression tree, with rules Θ regarding tree depth, stopping rules, etc...
- X_i is the data.
- c_b is the predicted value in each of the B regions.
- $I()$ is an indicator function.
- R_b defines the different regions in the space.

Stopping Rules

- Candidate splits must increase R^2 by a per-specified amount (the `cp` parameter, default=.01).
- Each candidate split must have at least `minsplit` observations in it (default=20).
- Each terminal node must have at least `minbucket` observations in it. Defaults to `round(minsplit/3)`.
- Tree depth - starting with the root node (0), what is the maximum depth of any node (defaults to 30).

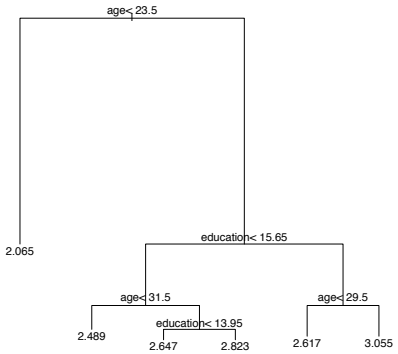
Example

```
library(rpart)
library(car)
data(SLID)
SLID <- SLID[complete.cases(SLID[,c("wages", "age", "education")]), ]
mod <- rpart(log(wages) ~ age + education, data=SLID)
mod
```

```
## n= 4014
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 4014 1018.09900 2.619255
##    2) age< 23.5 615   64.49511 2.064677 *
##    3) age>=23.5 3399  730.23310 2.719598
##      6) education< 15.65 2536  482.80130 2.641571
##       12) age< 31.5 554   85.64258 2.488905 *
##       13) age>=31.5 1982  380.63750 2.684244
##         26) education< 13.95 1560  288.89010 2.646745 *
##         27) education>=13.95 422   81.44458 2.822866 *
##      7) education>=15.65 863  186.62170 2.948886
##       14) age< 29.5 209   37.94680 2.617102 *
##       15) age>=29.5 654  118.31580 3.054915 *
```

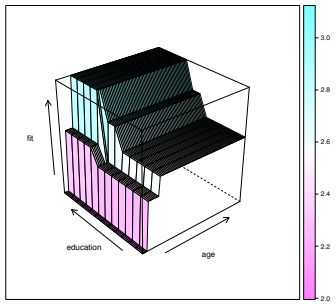

Decision Tree

```
plot(mod)
text(mod)
```



Surface Plot

```
age.s <- 20:95
educ.s <- 8:20
eg <- expand.grid(age=age.s, education=educ.s)
p <- predict(mod, newdata=eg)
eg$fit <- p
library(lattice)
wireframe(fit ~ age + education, data=eg, drape=T)
```



Compare to Other Models

```
library(mgcv)
library(KRLS)
lm.mod <- lm(log(wages) ~ education*age, data=SLID)
gam.mod <- gam(log(wages) ~ ti(education) + ti(age) +
  ti(age, education), data=SLID)
# krls.mod <- krls(log(wages) ~ education*age, data=SLID, derivative=F)
preds <- dget("http://quantoid.net/files/reg3/predictions.txt")
r <- do.call(cor, preds)^2
colnames(r) <- "R-squared"
r
```

```
##      R-squared
## [1,] 0.2618546
## [2,] 0.3577439
## [3,] 0.3570470
## [4,] 0.3352958
```

Smoothness of CART Models

Problems with CART

- Not particularly smooth - step functions aren't great approximations for smooth curves (though they can do OK).
- No real means for inference here. Bootstrapping can be problematic because the function is "non-regular" (small data changes can result in wild changes in the model)
- In more complicated models, it's difficult to figure out what effects look like.

Ensemble Methods

Ensemble methods produce a bunch of trees to better fit $f(X)$ and to prevent overfitting, with general form:

$$f(X_i) = \sum_{m=1}^{M} T_m(X_i, \Theta_m)$$

where m refers to the different trees.

- Random Forests (and tree bagging)
- Gradient boosting machines (GBM, including xgboost)
- Bayesian Additive Regression Trees (BART)

Tree Bootstrap Aggregating (Bagging)

The idea behind bagging:

- Draw lots of random samples from the data
- Fit a "deep" tree to each random sample.
- Average across the trees $\hat{f}_{\text{bag}}(X_i) = \frac{1}{M} \sum_{m=1}^{M} T_m(X_i, \Theta_m)$

Depends on the assumption of independence across trees to reduce bias and variance.

- Often a dubious assumption as trees are generally quite highly correlated.

Random Forests

A tree-bagging algorithm meant to increase independence across trees.

- In each random sample, only a small random subset (a) of the total j covariates is used in the splitting algorithm.
- Reduces bias and variance in the aggregate when a is small.

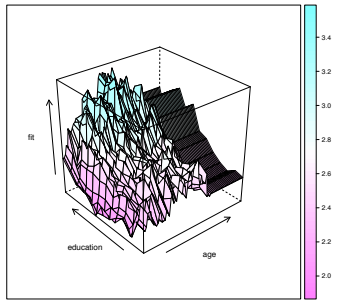
RandomForest in R

```
library(randomForest)
rfmod <- randomForest(log(wages) ~ age+education, data=SLID)
preds$x <- cbind(preds$x, rf=rfmod$predicted)
r <- do.call(cor, preds)
colnames(r) <- "R-squared"
r
```

```
##    R-squared
##    0.5117173
##    0.5981170
##    0.5975341
##    0.5790473
## rf 0.5558857
```


Surface

```
eg <- expand.grid(age=age.s, education=educ.s)
p <- predict(rfmod, newdata=eg)
eg$fit <- p
wireframe(fit ~ age + education, data=eg, drape=T)
```



Tree Boosting

Multiple tree method whereby trees are created sequentially

- Each new tree improves upon the prediction of the previous trees in the ensemble.
- Each new tree tries to accommodate observations that were not well predicted previously in the ensemble.

Gradient Boosting Machines (sum of trees model as above where):

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L\left(y_i, f_{m-1}(X_i) + T_m(X_i, \Theta_m)\right)$$

Here, f_{m-1} is the sum of trees up to the most recent one.

- Regularization by predicting a bunch of "weak" trees (small, simple trees that are weak predictors, though better than random).

Regularization in GBMs

Regularization in GBMs comes from two parameters

- B - Number of terminal nodes in each tree.
- ν - scales the contribution of each new tree to the fit

$$f(X_i) = f_{m-1}(X_i) + \nu T_m(X_i, \Theta_m)$$

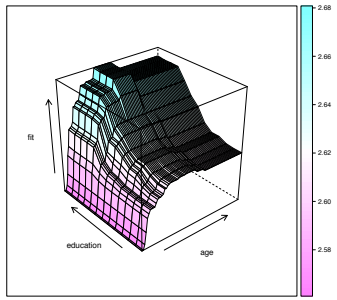
- Often times the value $\nu = 0.001$ is chosen.

GBMs in R

```
library(gbm)
gbm.mod <- gbm(log(wages) ~ age+education, data=SLID,
  interaction.depth=10)


## Distribution not specified, assuming gaussian ...


preds$x <- cbind(preds$x, GBM = gbm.mod$fit)
r <- do.call(cor, preds)
colnames(r) <- "R-squared"
r


##     R-squared
##     0.5117173
##     0.5981170
##     0.5975341
##     0.5790473
## rf  0.5558857
## GBM 0.6032400
```


Surface

```
eg <- expand.grid(age=age.s, education=educ.s)
p <- predict(gbm.mod, newdata=eg, n.trees=100)
eg$fit <- p
wireframe(fit ~ age + education, data=eg, drape=T)
```



Training in GBMs

GBMs can be useful, but there is often a non-trivial amount of training that has to be done in order to make them good predictors.

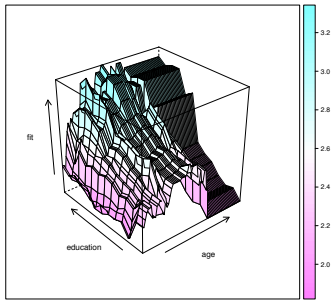
- Tuning these models requires more nuanced understanding on the part of the researcher.
- The `xgboost` algorithm is a different way of fitting these models that tends to produce better results with less tuning.

GBMs in R

```
library(xgboost)
form <- as.formula(log(wages) ~ age+education)
X <- model.matrix(form, data=SLID)[,-1]
y <- model.response(model.frame(form, SLID))
# xgb.mod <- xgboost(X,label=y, subsample=.5, nrounds=100)
# plot(1:100, as.vector(xgb.mod$evaluation_log[,2])[[1]],
#    type="l")
xgb.mod <- xgboost(X,label=y, subsample=.5, nrounds=10)


## [1] train-rmse:1.556568
## [2] train-rmse:1.129470
## [3] train-rmse:0.840781
## [4] train-rmse:0.654354
## [5] train-rmse:0.539842
## [6] train-rmse:0.470856
## [7] train-rmse:0.433848
## [8] train-rmse:0.413896
## [9] train-rmse:0.402292
## [10] train-rmse:0.395958


xgb.fit <- predict(xgb.mod, newdata=X)
preds$x <- cbind(preds$x, XGB = xgb.fit)
r <- do.call(cor, preds)
colnames(r) <- "R-squared"
r


##     R-squared
##     0.5117173
##     0.5981170
##     0.5975341
##     0.5790473
## rf  0.5558857
## GBM 0.6032400
## XGB 0.6289181
```

Surface

```
eg <- expand.grid(age=age.s, education=educ.s)
p <- predict(xgb.mod, newdata=as.matrix(eg), nrounds=10)
eg$fit <- p
wireframe(fit ~ age + education, data=eg, drape=T)
```



CV for XGBoost

Alternatively, we could use cross-validation to choose the parameters of the XGBoost model:

```
rsamp <- sample(1:nrow(X), floor(nrow(X)/2), replace=F)
dtrain <- xgb.DMatrix(X[rsamp, ], label = y[rsamp])
cv1 <- xgb.cv(data = dtrain, nrounds = 10, nthread = 2,
    nfold = 5, metrics = list("rmse"),
    max_depth = 3, eta = 1, objective = "reg:linear")
```

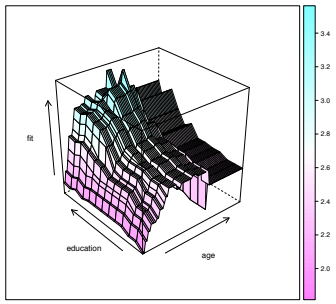
```
## [1] train-rmse:0.409409+0.005147 test-rmse:0.413451+0.020924
## [2] train-rmse:0.400421+0.005287 test-rmse:0.408450+0.023966
## [3] train-rmse:0.395596+0.005774 test-rmse:0.408740+0.023839
## [4] train-rmse:0.393297+0.004956 test-rmse:0.409157+0.023851
## [5] train-rmse:0.390602+0.004693 test-rmse:0.410518+0.024184
## [6] train-rmse:0.388681+0.004813 test-rmse:0.411871+0.023647
## [7] train-rmse:0.386189+0.005096 test-rmse:0.411965+0.023957
## [8] train-rmse:0.384367+0.004829 test-rmse:0.413613+0.024690
## [9] train-rmse:0.382652+0.004975 test-rmse:0.414173+0.025045
## [10] train-rmse:0.381333+0.004739 test-rmse:0.414835+0.025432
```

```
cv2 <- update(cv1, eta=.5)
```

```
## [1] train-rmse:1.152081+0.002439 test-rmse:1.154129+0.018252
## [2] train-rmse:0.674132+0.002442 test-rmse:0.678608+0.018817
## [3] train-rmse:0.482248+0.003107 test-rmse:0.489489+0.016364
## [4] train-rmse:0.418455+0.004166 test-rmse:0.427613+0.014099
## [5] train-rmse:0.399180+0.004195 test-rmse:0.411375+0.013212
## [6] train-rmse:0.393271+0.004069 test-rmse:0.407082+0.013317
## [7] train-rmse:0.390627+0.003738 test-rmse:0.405796+0.013774
## [8] train-rmse:0.388761+0.003978 test-rmse:0.405162+0.013798
## [9] train-rmse:0.386883+0.004274 test-rmse:0.405765+0.013787
## [10] train-rmse:0.385752+0.004585 test-rmse:0.406581+0.014136
```


Surface

```
age.s <- 20:95
educ.s <- 8:20
eg <- expand.grid(age=age.s, education=educ.s)
p <- predict(xgb.mod2, newdata=as.matrix(eg), nrounds=8)
eg$fit <- p
wireframe(fit ~ age + education, data=eg, drape=T)
```



Bayesian Additive Regression Trees (BART)

Similar to GBMs in that each tree investigates residuals from previous trees with regularization.

$$y_i = \sum_{m=1}^{M} T_m(X_i, \Theta_m) + \epsilon_i \qquad \epsilon_i \sim N(0, \sigma^2)$$

- Tree growing and regularization are done via MCMC (which provide a means for inference)
- Independent priors placed over number of trees, tree depth, variables used in (and corresponding values of) splits and the error variance.
- As with GBMs, regularization results in no one tree being able to dominate the fit.
- Tuning parameters should be chosen via cross-validation.

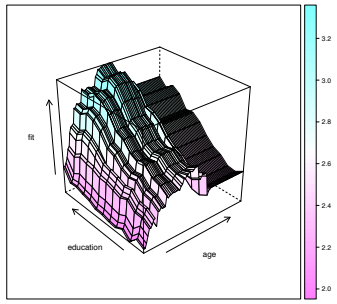
BART in R

```
library(rJava)
library(bartMachine)
X <- model.matrix(log(wages) ~ age+education, data=SLID)[,-1]
y <- model.response(model.frame(log(wages) ~ age+education, data=SLID))
bart.mod <- bartMachine(as.data.frame(X),y, verbose=F)
preds$x <- cbind(preds$x, BART = bart.mod$y_hat_train)
r <- do.call(cor, preds)
colnames(r) <- "R-squared"
r
```

```
##      R-squared
##      0.5117173
##      0.5981170
##      0.5975341
##      0.5790473
## rf   0.5558857
## GBM  0.6032400
## XGB  0.6289181
## BART 0.6133894
```


Surface

```
eg <- expand.grid(age=age.s, education=educ.s)
pred.bart <- predict(bart.mod, new_data=eg)
eg$fit <- pred.bart
wireframe(fit ~ age + education, data=eg, drape=T)
```



'Real" Example

```
library(readstata13)
banks <- read.dta13("http://quantoid.net/files/reg3/banks99.dta")
banks.dat <- banks[,-c(1,2,4,5)]
X <- model.matrix(gdppc_mp ~ ., data=banks.dat)
y <- model.response(model.frame(gdppc_mp~ ., data=banks.dat))
cart1 <- rpart(y ~ X)
rf1 <- randomForest(X,y)
gbm1 <- gbm(gdppc_mp ~ ., data=banks.dat, interaction.depth=10)
rsamp <- sample(1:nrow(X), floor(nrow(X)/2), replace=F)
xgbc <- xgb.cv(data=X[rsamp, ], label=y[rsamp], max_depth=5,
    eta=.5, nrounds=25, nfold=3)
xgb1 <- xgboost(data = X, label=y, subsample=.5, max_depth=5,
    eta=.5, nrounds=10)
bart1 <- bartMachine(as.data.frame(X),y, verbose=F)
```

```
preds <- cbind(
  CART = predict(cart1, newdata=as.data.frame(X)),
  RF = rf1$predicted, GBM = gbm1$fit,
  XGB = predict(xgb1, newdata=X),
  BART = bart1$y_hat_train)
cor(preds, y)^2
```

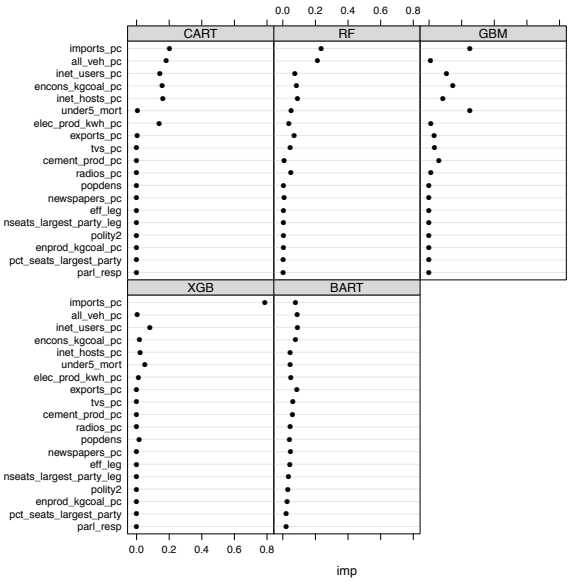
```
##           [,1]
## CART 0.9291945
## RF   0.8436942
## GBM  0.5722043
## XGB  0.9868310
## BART 0.9997247
```


Evaluating Tree Models: Variable Importance

How do we know which variables are important in the models we fit?

- CART: Sum of the goodness of split measures for each split for which it was the primary variable, plus goodness * (adjusted agreement) for all splits in which it was a surrogate.
- RF: 1) Difference in MSE decrease relative to random permutation of variable, 2) Total decrease in residual sum of squares.
- GBM: Same as number 1 for Random Forests
- XGBOOST: Gain - relative contribution of feature to model (based on the gradient of each feature), Cover - relative # observations related to each feature, Frequency - percentage of times a feature occurs in the trees of the model.
- BART: Posterior probability of inclusion in 1) splits or 2) trees.

Importance Plot



Visualizing Partial Effects: Partial Dependence Plot

The PDP plots the change in the average predicted value for a subset of features S, averaged over the subset of features C, where C is the complement of S. Formally:

$$f_S = \mathbb{E}_{x_C}\left[f(\boldsymbol{x}_S, \boldsymbol{x}_C)\right] = \int f(\boldsymbol{x}_S, \boldsymbol{x}_C) dP(\boldsymbol{x}_C)$$

In words: we are predicting $f()$ with the variables in S averaged over all of the variables in C.

Visualizing Partial Effects: Individual Conditional Expectation Plots

ICE disaggregates the PDP.

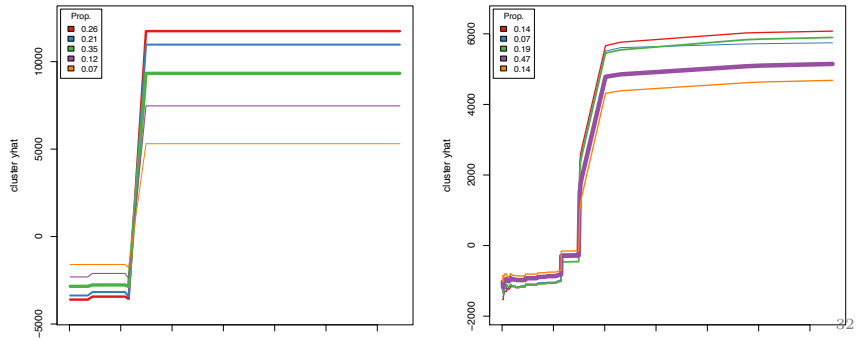
- The PDP is obtained by averaging over all of the ICE curves.
- Plots N different curves to enable evaluation of effect heterogeneity.
- Heterogeneity essentially means interactions with variables in C.

$$f_{S_i} = \mathbb{E}_{x_{C_i}}\left[f(\boldsymbol{x}_S, \boldsymbol{x}_{C_i})\right]$$

Ice Ice Baby

```
library(RColorBrewer)
cols <- brewer.pal(5, "Set1")
library(ICEbox)
ice1 <- ice(xgb1, X=X, y=y, predictor="imports_pc",
  nrounds=10)
cice1 <- clusterICE(ice1, nClusters=5, plot_legend=TRUE,
  colorvec=cols)
```

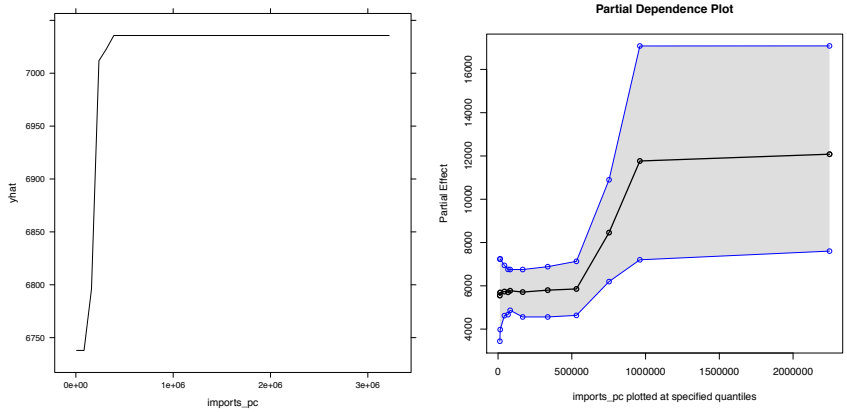
```
ice2 <- ice(bart1, as.data.frame(X), y, predictor="imports_pc")
cice2 <- clusterICE(ice2, nClusters=5, plot_legend=TRUE,
  colorvec=cols)
```



PDPs in R

```
library(pdp)
p1 <- partial(gbm1, pred.var="imports_pc",
  n.trees=100, chull=TRUE)
plotPartial(p1)
```

```
p2 <- pd_plot(bart1, j="imports_pc")
```

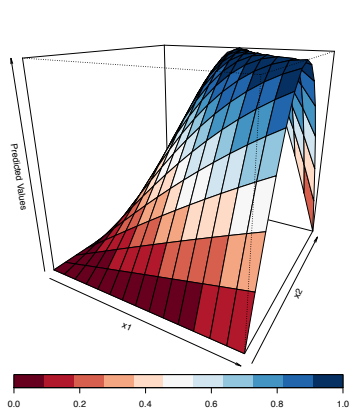


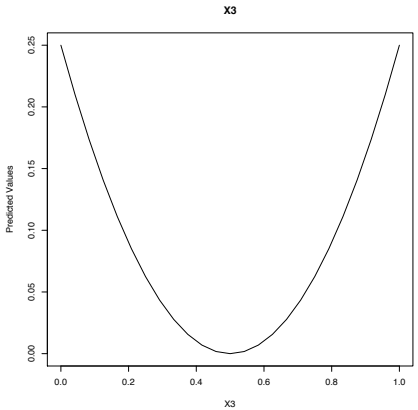
Friedman's Data

```
set.seed(11)
n  = 200
p = 5
X = data.frame(matrix(runif(n * p), ncol = p))
Xm <- as.matrix(X)
y = 10 * sin(pi* X[ ,1] * X[,2]) +20 *
  (X[,3] -.5)^2 + 10 * X[ ,4] + 5 * X[,5] + rnorm(n)
df <- as.data.frame(cbind(X,y))
cart2 <- rpart(y ~ ., data=df)
rf2 <- randomForest(y ~ ., data=df)
gbm2 <- xgboost(data=Xm, label=y, nrounds=5, subsample=.5)


## [1] train-rmse:10.734185
## [2] train-rmse:8.022166
## [3] train-rmse:6.076094
## [4] train-rmse:4.707265
## [5] train-rmse:3.682009


bart2 <- bartMachine(X, y, verbose=F)
```


Friedman True Relationships

`X4` and `X5` are linear and additive in the equation.





Friedman ICE Plots

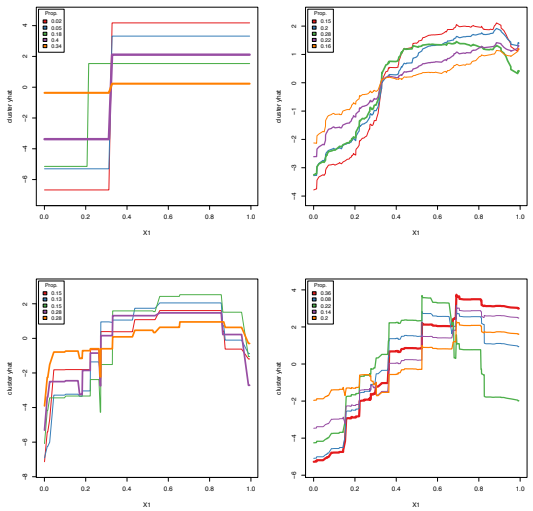
```
library(RColorBrewer)
cols <- brewer.pal(5, "Set1")
fice1 <- ice(cart2, X=X, y=y, predictor="X1")
clusterICE(fice1, nClusters=5, plot_legend=TRUE,
  colorvec=cols)
```

```
fice2 <- ice(rf2, X=X, y=y, predictor="X1", num.trees=100)
clusterICE(fice2, nClusters=5, plot_legend=TRUE,
  colorvec=cols)
```

```
fice3 <- ice(gbm2,  X=Xm, y=y, predictor="X1", n.trees=100)
clusterICE(fice3, nClusters=5, plot_legend=TRUE,
  colorvec=cols)
```

```
fice4 <- ice(bart2, X=X, y=y, predictor="X1")
clusterICE(fice4, nClusters=5, plot_legend=TRUE,
  colorvec=cols)
```

ICEs



Friedman PDPs

```r
fp1 <- partial(cart2, train=df, pred.var="X1")
plotPartial(fp1)
```

```r
fp2 <- partial(rf2,  pred.var="X1", num.trees=100)
plotPartial(fp2)
```

```r
xgb.plot.shap(model=gbm2, data=Xm, features=c("X1"))
```

```r
pd_plot(bart2,  j="X1")
```


PDPs