Answers Head Start data

Marjolein Fokkema

Load data and separate train and test observations:

```
HSdata <- readRDS(file = "HeadStart.Rda")
set.seed(42)
train_ids <- sample(1:nrow(HSdata), size = nrow(HSdata)/2)
```

Exercise 1: Fit a generalized additive model with smoothing splines

a)

```
library("mgcv")
HSdata <- readRDS(file = "HeadStart.Rda")
set.seed(42)
gamod <- gam(Test_Pct04 ~ s(AgeAFQT) + s(PermInc) + Race_Child +
               s(HOME_Pct88) + Father_HH86,
             data = HSdata[train_ids,], distribution = "gaussian")
summary(gamod)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Test_Pct04 ~ s(AgeAFQT) + s(PermInc) + Race_Child + s(HOME_Pct88) +
##     Father_HH86
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    56.392      2.789  20.220   <2e-16 ***
## Race_Child2    -5.047      2.308  -2.187   0.0289 *
## Race_Child3     3.544      2.110   1.680   0.0933 .
## Father_HH861    1.520      2.308   0.659   0.5103
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                 edf Ref.df      F p-value
## s(AgeAFQT)    2.729  3.431 16.327 < 2e-16 ***
## s(PermInc)    3.153  3.944  5.821 0.00015 ***
## s(HOME_Pct88) 1.000  1.000  5.012 0.02538 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.295   Deviance explained = 30.2%
## GCV = 491.96  Scale est. = 486.98    n = 1075
```

The effects are presented separately for predictors with parametric effects (specified without function `s()` in the model formula) and those with non-parametric effects (i.e., the smoothing splines, specified with function `s()` in the model formula).

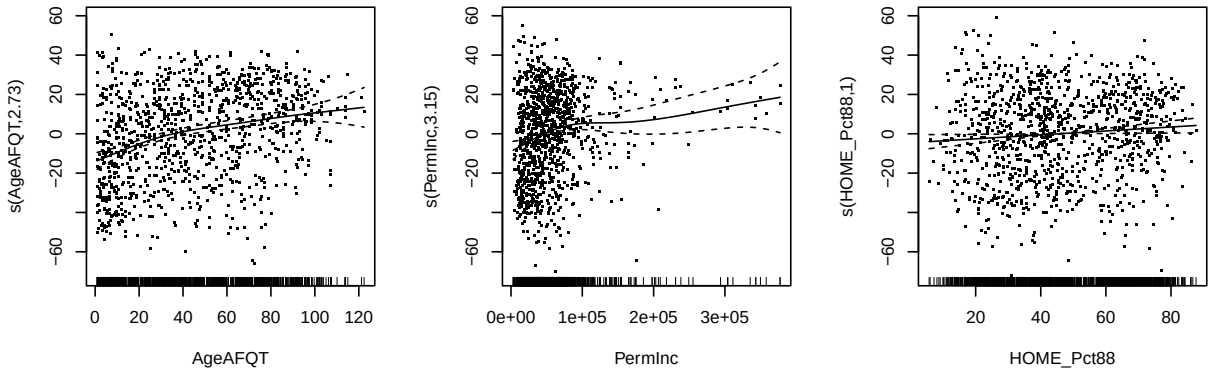
All five predictors have a significant effect on the outcome.

Compared to the reference category of Hispanic children who did not have their father living in the family household in 1986, Black children ('Race' = 2) have significantly lower cognitive test scores in 2004, and white children (`Race` = 3) seem to have somewhat higher performance, but this difference is non-significant. The father living in the family household in 1986 has a smaller but positive effect on the cognitive test score in 2004.

The three continuous predictors have stronger effects, judging by their F values. From this output, we cannot see the direction of their effects. The empirical degrees of freedom (`edf`) indivate to what extent the effects are non-linear. An `edf` value close to 1 indicates a linear effect, higher `edf` values indicate more non-linear effects. The percentile score for emotional and cognitive support at home has a linear effect, while the mother's intelligence test score, permanent family income have non-linear effects on the outcome.

b) Apply function `plot` to the fitted model and interpret the plots.

```
par(mfrow = c(1, 3))
plot(gamod, residuals = TRUE)
```



The mother's intelligence test score, permanent family income and the percentile score for emotional and cognitive support at home combined have a positive effect on the the cognitive test score in 2004. As already suggested by the `edf` values, the percentile score for emotional and cognitive support at home has a linear effect, while the mother's intelligence test score, permanent family income have non-linear effects on the outcome.

c)

```
## Benchmark: MSE for predicting the (training set) mean for all
var(HSdata$Test_Pct04[train_ids])
```

```
## [1] 690.7769
```

```
mean((HSdata$Test_Pct04[-train_ids] - mean(HSdata$Test_Pct04[train_ids]))^2)
```

```
## [1] 690.0514
```

```
## Test predictions and MSE
gam_preds_test <- predict(gamod, newdata = HSdata[-train_ids,])
mean((gam_preds_test - HSdata$Test_Pct04[-train_ids])^2)
```

```
## [1] 494.9123
```

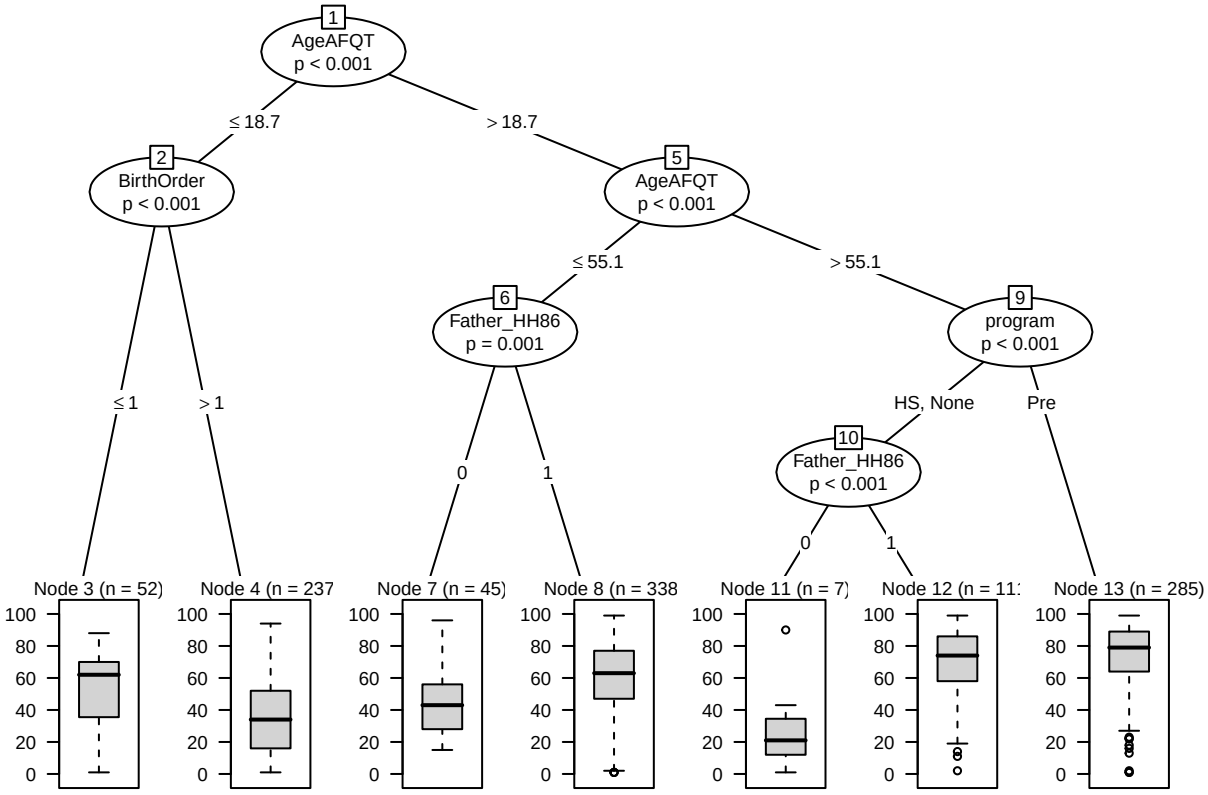
```
## Train predictions and MSE
gam_preds_train <- predict(gamod, newdata = HSdata[train_ids,])
mean((gam_preds_train - HSdata$Test_Pct04[train_ids])^2)
```

```
## [1] 482.0476
```

The MSE on training and test data are not too different, suggesting very little overfitting. About 28% of variance on test data is explained by the GAM.

Exercise 2: Fit a decision tree

```
library("partykit")
ct <- ctree(Test_Pct04 ~ ., data = HSdata[train_ids,])
plot(ct, gp = gpar(cex = .6))
```



The tree reveals positive effects of the mother's intelligence test score and presence of the father in the household on cognitive test performance. For children of mother's with lowest intelligence test scores, birth order has a negative effect, with later children having lower cognitive test scores. Furthermore, we see that going to pre-school has a positive effect, for children of mothers with higher intelligence test scores, compared to Head Start or no preschool.

b)

```
ct_test_preds <- predict(ct, newdata = HSdata[-train_ids,])
mean((ct_test_preds - HSdata$Test_Pct04[-train_ids])^2)
```

```
## [1] 527.8639
```

```
ct_train_preds <- predict(ct, newdata = HSdata[train_ids,])
mean((ct_train_preds - HSdata$Test_Pct04[train_ids])^2)
```

```
## [1] 490.247
```

The tree performs somewhat worse than the GAM, which is expected. It also uses one predictor less for prediction. There is a somewhat larger difference between train and test performance, suggesting more proneness to overfitting for the tree than for the GAM.

Exercise 3: Fit a bagged tree ensemble and random forest

a)

```
library("randomForest")
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(42)
bag <- randomForest(Test_Pct04 ~ ., mtry = ncol(HSdata)-1L,
                    data = HSdata[train_ids,], importance = TRUE)
```

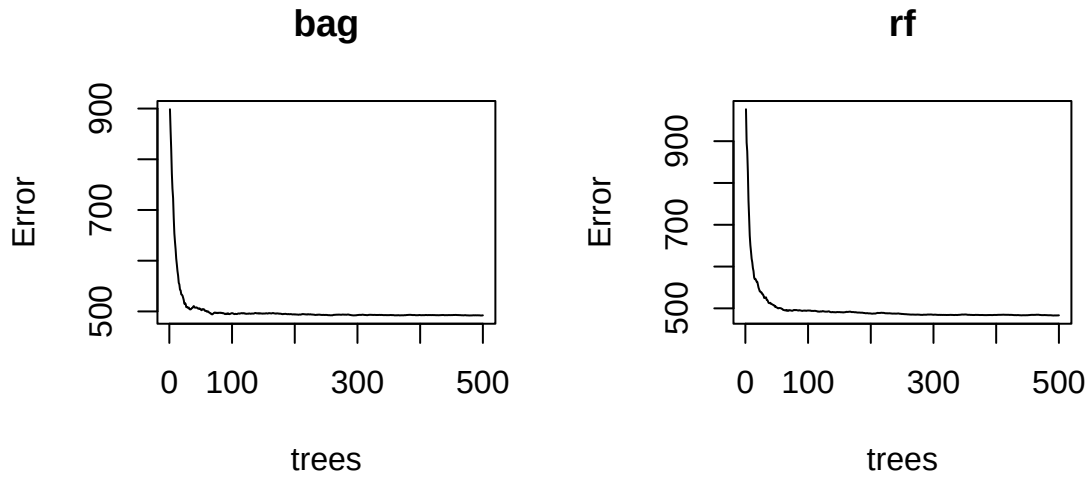
b)

```
library("randomForest")
set.seed(42)
rf <- randomForest(Test_Pct04 ~ ., data = HSdata[train_ids,], importance = TRUE)
```

c) By default, `ntree = 500`, thus 500 trees are generated. Arguments `nodesize` (defaults to 5 for classification and 1 for regression) and `maxnodes` (default is `NULL`, which results in no maximum for the number of nodes) control the depth of the fitted trees. By default, `replace = TRUE`, thus bootstrap sampling is employed. If we specify `replace = FALSE`, sub sampling will be employed (and a proportion of .632 of the observations will be selected for each sample, see also argument `sampsize`).

d)

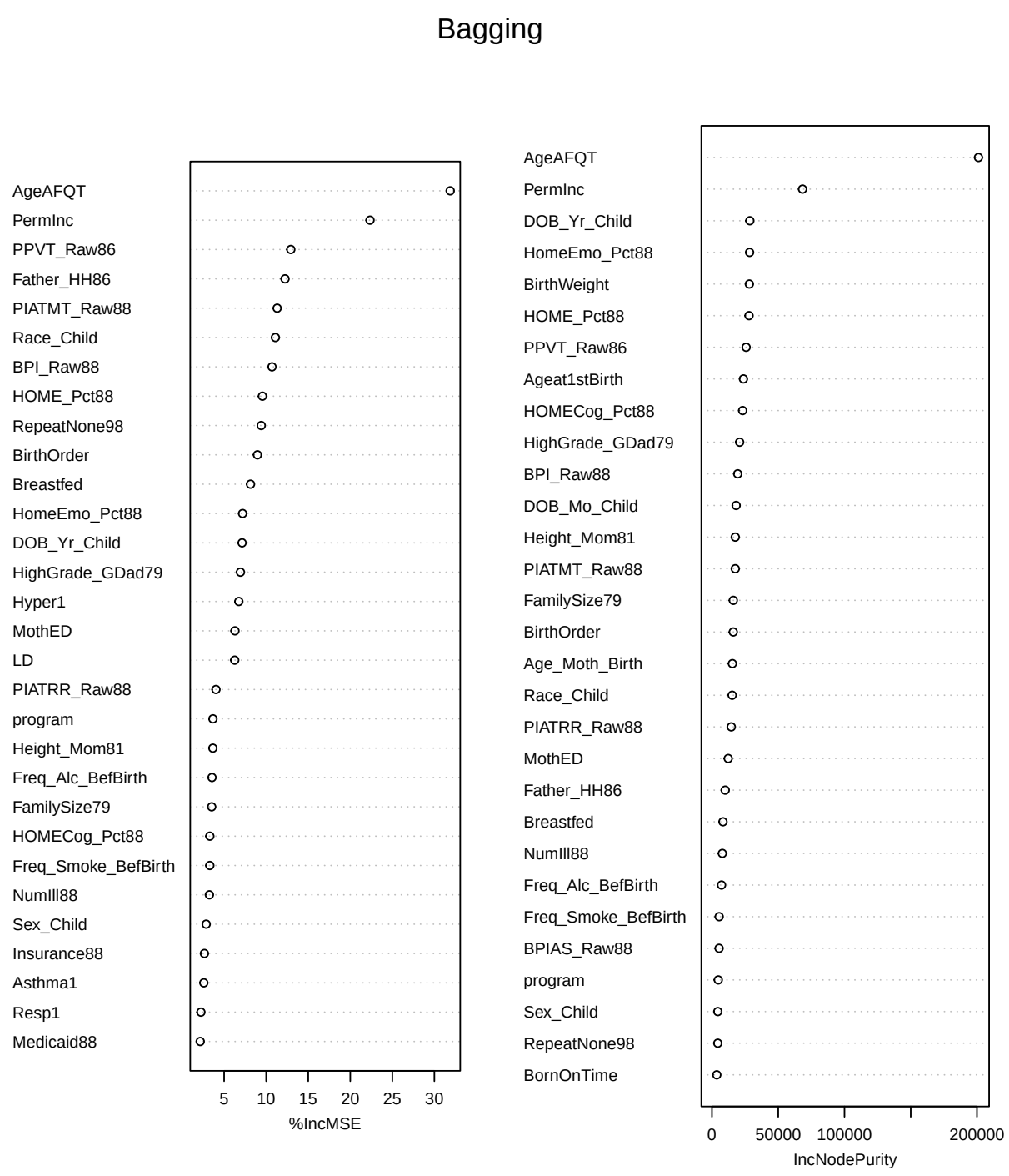
```
par(mfrow = c(1, 2))
plot(bag)
plot(rf)
```



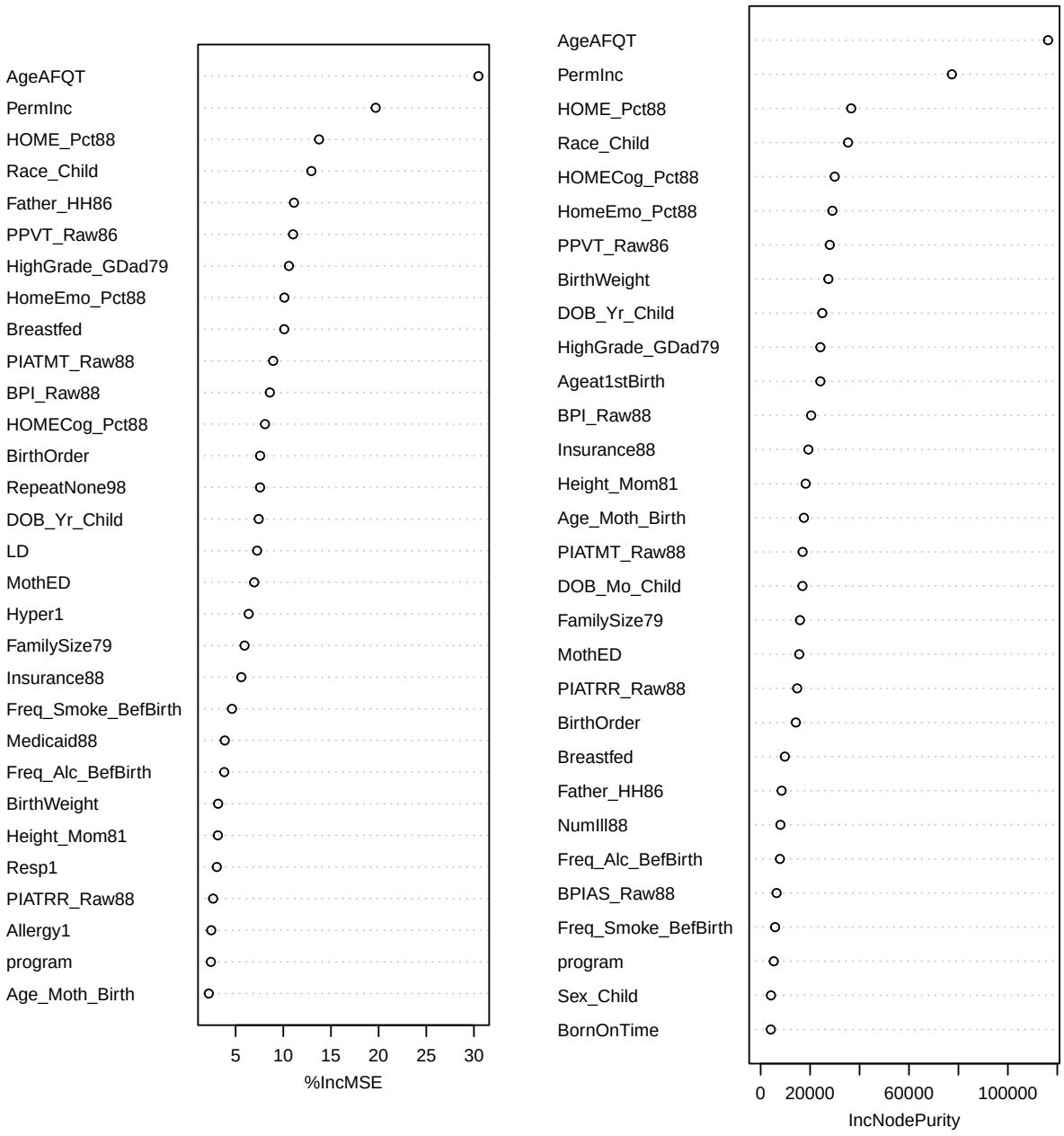
OOB error seems to have stabilized and is unlikely to increase or reduce by increasing the number of trees in any of the two ensembles.

e)

```
varImpPlot(bag, cex = .7, main = "Bagging")
```



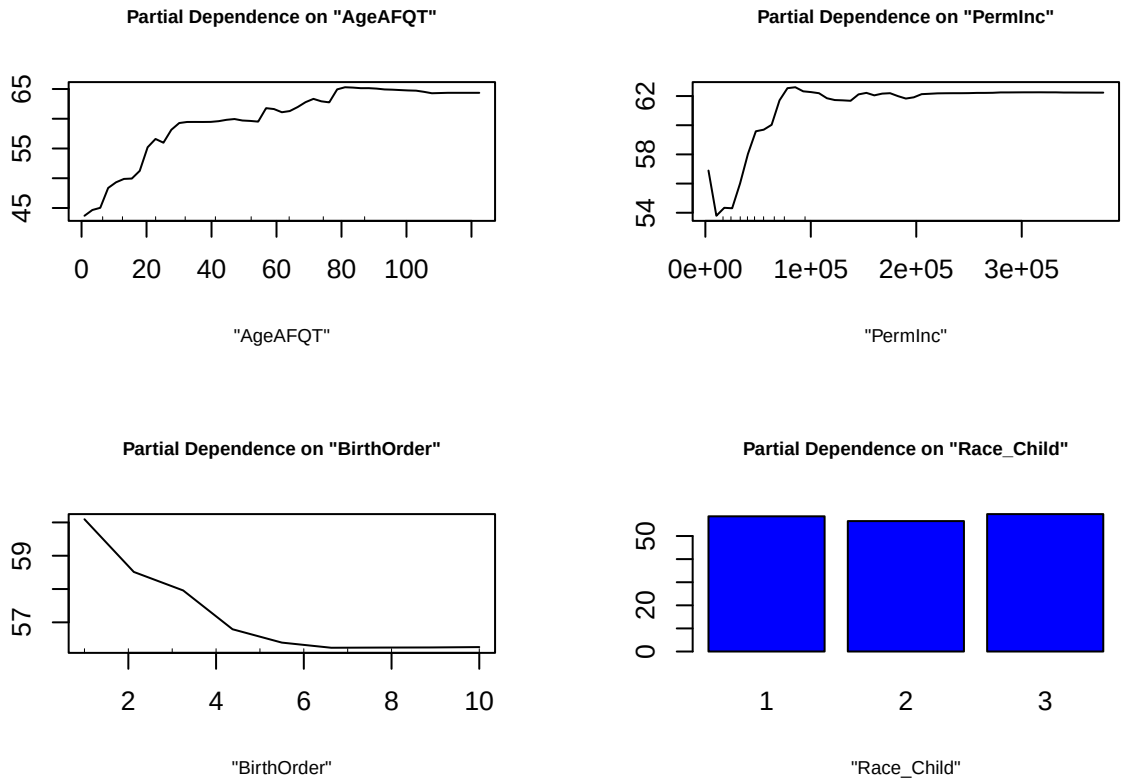

```
varImpPlot(rf, cex = .7, main = "Random Forest")
```



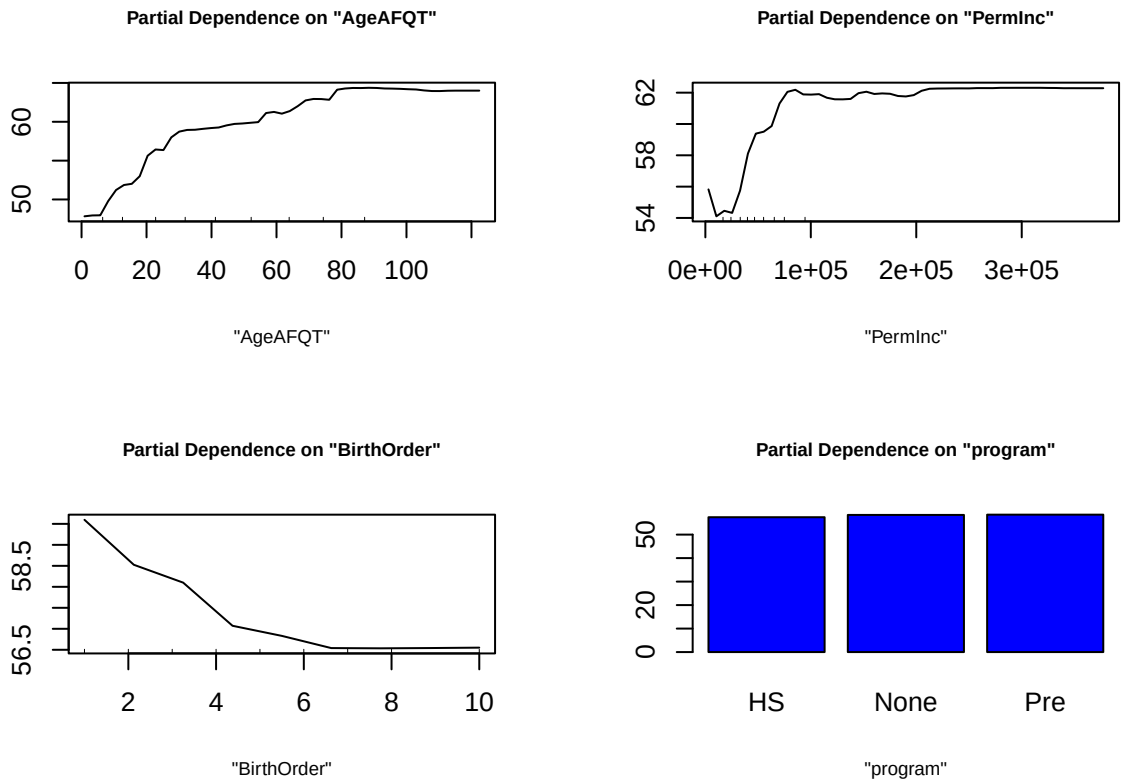
Both the bagged and RF ensemble have `AgeAFQT` and `PermInc` as the strongest predictors.

```
par(mfrow = c(2,2))
partialPlot(bag, x.var = "AgeAFQT", pred.data = HSdata[train_ids, ],
            cex.lab = .7, cex.main = .7)
partialPlot(bag, x.var = "PermInc", pred.data = HSdata[train_ids, ],
            cex.lab = .7, cex.main = .7)
partialPlot(bag, x.var = "BirthOrder", pred.data = HSdata[train_ids, ],
            cex.lab = .7, cex.main = .7)
```

```
partialPlot(bag, x.var = "Race_Child", pred.data = HSdata[train_ids, ],
            cex.lab = .7, cex.main = .7)
```




```
par(mfrow = c(2,2))
partialPlot(rf, x.var = "AgeAFQT", pred.data = HSdata[train_ids, ],
            cex.lab = .7, cex.main = .7)
partialPlot(rf, x.var = "PermInc", pred.data = HSdata[train_ids, ],
            cex.lab = .7, cex.main = .7)
partialPlot(rf, x.var = "BirthOrder", pred.data = HSdata[train_ids, ],
            cex.lab = .7, cex.main = .7)
partialPlot(rf, x.var = "program", pred.data = HSdata[train_ids, ],
            cex.lab = .7, cex.main = .7)
```



The partial dependence plots reveal highly similar effects as the GAM. The effect of the program (Head Start, preschool or no program) seem very minor, as was the case in the single tree.

g)

```
## Test error
bag_preds_test <- predict(bag, newdata = HSdata[-train_ids, ])
mean((bag_preds_test - HSdata$Test_Pct04[-train_ids])^2)
```

```
## [1] 485.629
```

```
rf_preds_test <- predict(rf, newdata = HSdata[-train_ids, ])
mean((rf_preds_test - HSdata$Test_Pct04[-train_ids])^2)
```

```
## [1] 483.0401
```

```
## Training error
bag_preds_train <- predict(bag, newdata = HSdata[train_ids, ])
mean((bag_preds_train - HSdata$Test_Pct04[train_ids])^2)
```

```
## [1] 79.12862
```

```
rf_preds_train <- predict(rf, newdata = HSdata[train_ids, ])
mean((rf_preds_train - HSdata$Test_Pct04[train_ids])^2)
```

```
## [1] 83.01596
```

The random forest performs better on test data than the bagged ensemble and both outperform the GAM and tree. Note that in spite of the strong overfitting behavior on the training data of the bagged ensemble and random forest (training MSE is very small compared to test MSE), the methods still perform best on test data. In fact, if we would not have specified the `newdata` argument, the `predict` function for `randomForest` would have returned OOB predictions, which do not overfit:

```
## out-of-bag error
bag_preds_oob <- predict(bag)
mean((bag_preds_oob - HSdata$Test_Pct04[train_ids])^2)
```

```
## [1] 492.2711
```

```
rf_preds_oob <- predict(rf)
mean((rf_preds_oob - HSdata$Test_Pct04[train_ids])^2)
```

```
## [1] 483.3273
```

The OOB errors provide somewhat more conservative estimates than the test error.

Exercise 4: Fit a gradient boosted tree ensemble

```
library("gbm")
```

Note that if you load the `gbm` package you'll get a message that you should consider using package `gbm3` instead, which is in fact no longer available, so you can ignore this message.

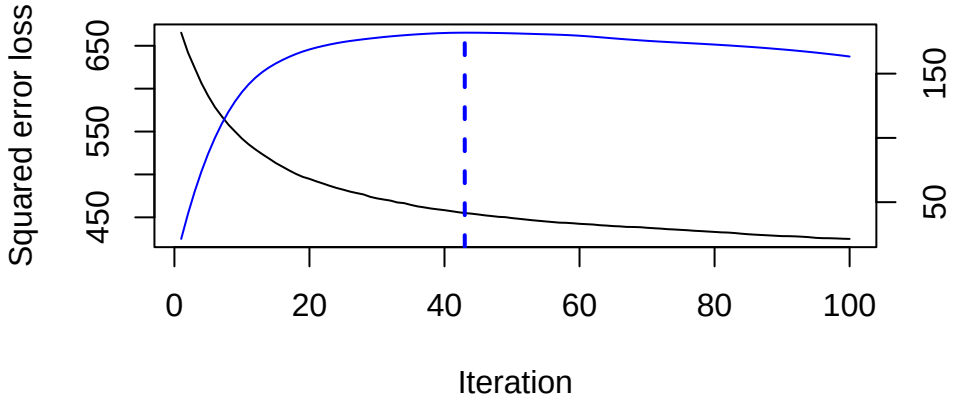
a) The learning rate is controlled by the `shrinkage` argument which defaults to a value of .1. The `n.trees` argument controls the number of generated trees, which defaults to 100. The `bag.fraction` argument controls the fraction of training set observations randomly generated to fit each tree in the ensemble and defaults to 0.5. Tree depth is controlled by argument `interaction.depth`, which defaults to 1 (trees with a single split and 2 terminal nodes, i.e., main effects only).

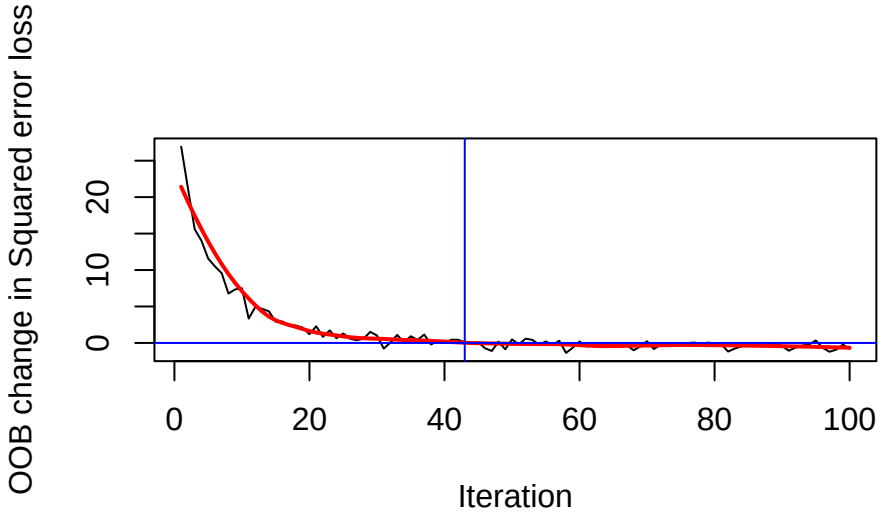
b)

```
set.seed(42)
boost.ens <- gbm(Test_Pct04 ~ ., data = HSdata[train_ids,],
                 distribution = "gaussian")
```

c)

```
par(mfrow = c(2, 1))
gbm.perf(boost.ens, oobag.curve = TRUE)
```






```
## [1] 43
## attr(,"smoother")
## Call:
## loess(formula = object$oobag.improve ~ x, enp.target = min(max(4,
##     length(x)/10), 50))
##
## Number of Observations: 100
## Equivalent Number of Parameters: 8.32
## Residual Standard Error: 0.9622
```

First plot: The black curve depicts training error as a function of the number of trees. In general, it can only decrease. The blue curve corresponds to the 2nd y axis on the right, reflecting the cumulative improvement in loss (so it is more or less the exact opposite of the black curve). The blue vertical dotted line gives the iteration where OOB error starts to deteriorate. The printed message (suppressed here) notes that `OOB generally underestimates the optimal number of iterations` so we should take this approximation with a grain of salt.

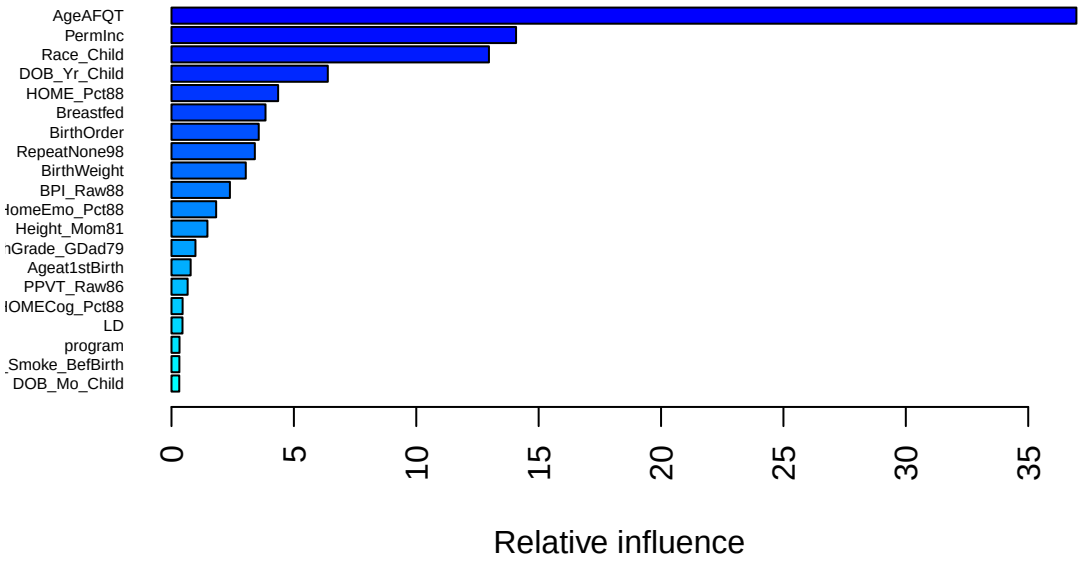
The second plot gives a curve for the improvement in loss from one iteration or tree to the next. The change is positive for the first trees, the vertical line is drawn at a change of 0, and the change becomes negative afterwards. This is where the location of the vertical line in the previous plot comes from. The thick black lines gives the empirical estimates, the red curve is a smoothened version of that.

From these plots, we can conclude that given the learning rate of .01, we have fitted enough trees. We might want to use 100 of the trees, or slightly less, for prediction.

Also see `?gbm.perf` for an explanation of what these plots show

d)

```
imps <- summary(boost.ens, cBars = 20, las = 2, cex.names = .5)
```

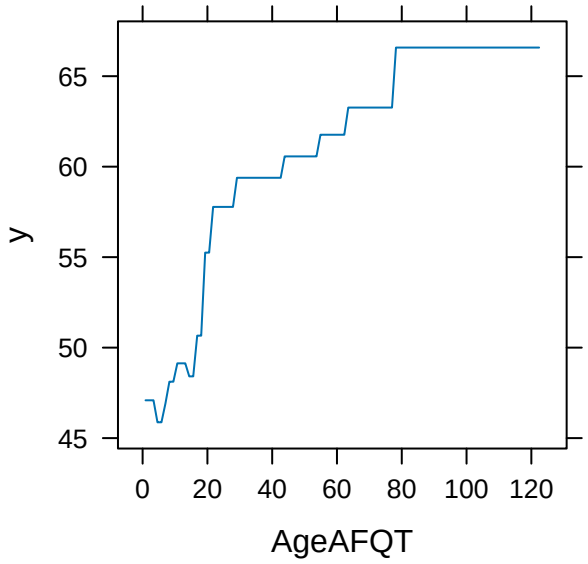


The most important predictors are `AgeAFQT`, `PermInc` and `Race_Child`.

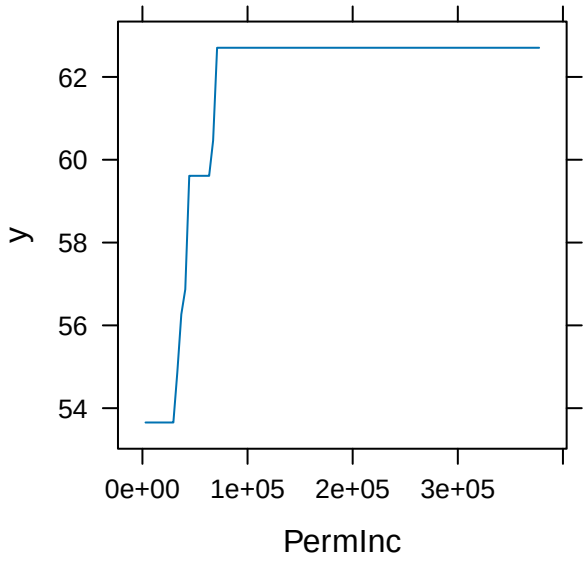
e) By default, function `relative.influence` is used for computing importances, which does not use permutation, permutation would be used if we specify `method = permutation.test.gbm`. Because gradient boosted ensembles do not really have OOB observations. Trees are fitted on samples of the training data, but because of the sequential nature of gradient boosting, observations not part of the sample still influence the tree, so are not really OOB anymore.

f)

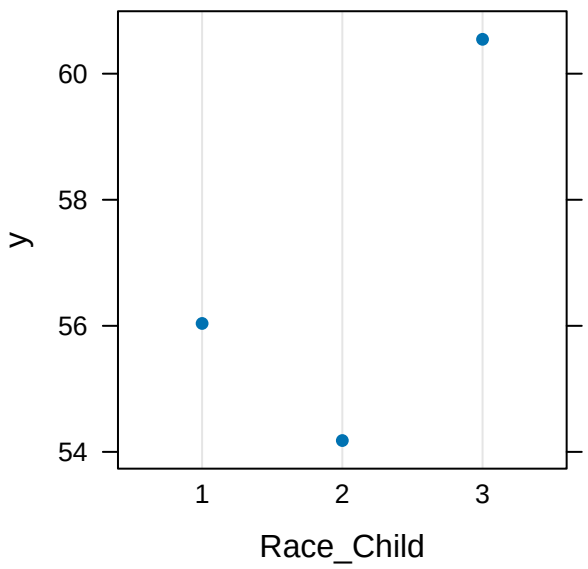
```
plot(boost.ens, i.var = "AgeAFQT")
```




```
plot(boost.ens, i.var = "PermInc")
```




```
plot(boost.ens, i.var = "Race_Child")
```



The PDPs display a positive effect of the mother's intelligence test score `AgeAFQT` and of the averaged family income `PermInc`. Further, Hispanic children (`Race_Child = 1`) appear to score higher on the 2004 test than White (`Race_Child = 1`) and Black (`Race_Child = 2`) and White.

g)

```
gbm_preds_train <- predict(boost.ens, newdata = HSdata[train_ids, ])
```

```
## Using 100 trees...
```

```
mean((gbm_preds_train - HSdata$Test_Pct04[train_ids])^2)
```

```
## [1] 424.8655
```

```
gbm_preds_test <- predict(boost.ens, newdata = HSdata[-train_ids, ])
```

```
## Using 100 trees...
```

```
mean((gbm_preds_test - HSdata$Test_Pct04[-train_ids])^2)
```

```
## [1] 478.8546
```

The gradient boosting ensemble outperforms the single tree as well as the bagging and random forest ensembles on test data.

Note also how the gradient boosting ensemble has much less overoptimism on training data. Small trees are much more regularized and tend to overfit much less than the large trees in bagging and random forests.

Exercise 5: Tune parameters of a gradient boosted tree ensemble

Load library, specify grid and prepare data:

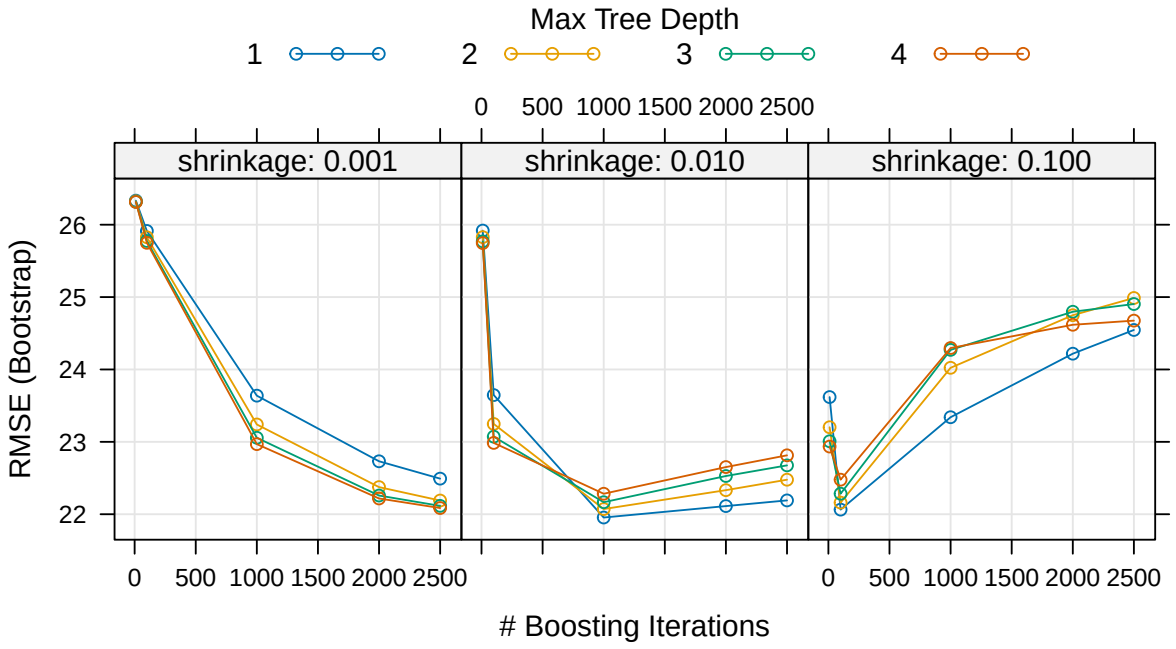
```
library("caret")
grid <- expand.grid(shrinkage = c(.1, .01, .001),
                    n.trees = c(10, 100, 1000, 2000, 2500),
                    interaction.depth = 1:4,
                    n.minobsinnode = 10)
y <- HSdata$Test_Pct04[train_ids]
x <- HSdata[train_ids, -which(names(HSdata) == "Test_Pct04")]
```

a) Apply function `train` (and wait patiently, a lot of models are fitted and evaluated):

```
set.seed(42)
gbmFit <- train(x = x, y = y, tuneGrid = grid,
                distribution = "gaussian", method = "gbm",
                trControl = trainControl(number = 10L, verboseIter = TRUE))
```

b) Print and plot the result:

```
#print(gbmFit) ## omitted for space considerations
plot(gbmFit)
```




```
gbmFit$bestTune
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 23    1000                 1      0.01             10
```

For `shrinkage` of .1 and .01, we obtain best performance with a tree depth of 1, suggesting main effects only and no interactions. Although tree depth of four seems to do better for a learning rate of .001, lower tree size might do better if we would try a larger number of trees (i.e., increase the number of boosting iterations). If we want to perfectly optimize predictive accuracy, we might try numebr of trees > 2500 with the learning rate of .001. For this exercise, I think this optimized set of hyperparameters is good enough.

c) The main effect of `shrinkage`, `n.trees` and `interaction.depth` are difficult to describe, because there appear to be interaction(s).

There appears to be an interaction between tree depth and number of trees. With lower number of trees, larger tree depth is needed. With higher number of trees, smaller tree depth is needed. This is quite typical in (gradient boosted) tree ensembles. With smaller trees, the bias is high, so many trees are needed to reduce bias.

Only for the higher shrinkage values of .1 and .01 does this effect kick in, in the grid of hyper parameters tried. Probably, with shrinkage value of .001, an even larger number of trees is needed for this effect to kick in.

d) Given the good performance of `interaction.depth` of 1 at `shrinkage` of .1 and .01, there do not appear to be (strong) interaction effects in the data.

e) Refit the ensemble using optimal parameter values:

```
set.seed(42)
boost.ens2 <- gbm(Test_Pct04 ~ ., data = HSdata[train_ids, ], distribution = "gaussian",
            n.trees = 1000, interaction.depth=1, shrinkage = 0.01)
gbm_preds_test2 <- predict(boost.ens2, newdata = HSdata[-train_ids, ])
mean((gbm_preds_test2 - HSdata$Test_Pct04[-train_ids])^2)
```

```
## [1] 473.9186
```

```
gbm_preds_train2 <- predict(boost.ens2, newdata = HSdata[train_ids, ])
mean((gbm_preds_train2 - HSdata$Test_Pct04[train_ids])^2)
```

```
## [1] 423.5884
```

The predictive performance of the boosting model was further improved by tuning its parameters. The difference between training and test MSE is still much smaller than for the bagged ensemble and random forest, illustrating that gradient boosting overfits much less. The tuned gradient boosting model explains about 39% of variance on the test data, while the worst performing method (single tree) explained about 23% of variance on the test data.

It should be noted, however, that for prediction, the tree ensembles use all 51 predictors, while the GAM and the tree only used five and four predictors, respectively.