Bagging

The idea of **bootstrap aggregation**, or **bagging**, is to generate a prediction from each of B bootstrap samples and average them.

It is an **ensemble method**.

Such methods are useful when combined with methods like CART, which are nonlinear and have high variance.

- CART has high variance because small changes in the sample can greatly affect the locations of the splits.
- Hence the predictions can change a lot when we add or subtract a few observations.

This is much less true for linear regression, unless some observations happen to be very influential.

Unless p is large relative to n, modest changes in a sample tend to have modest effects on the predictions from linear regression.

Let $\hat{f}_b^*(x_0)$ be the prediction for the point x_0 based on bootstrap sample b. Then the bagging estimate is

$$\hat{f}_{\text{bag}}(x_0) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_b^*(x_0). \tag{1}$$

This is an approximation to $\mathrm{E}\big(\hat{f}_b^*(x_0)\big)$ based on the empirical distribution of the points (x_i, y_i).

If we let $B \to \infty$, then $\hat{f}_{\text{bag}}(x_0)$ would converge to

$$\hat{f}_{\text{ag}}(x_0) = \mathrm{E}\big(\hat{f}^*(x_0)\big), \tag{2}$$

where the expectation is taken with respect to the EDF of the data.

The **empirical distribution** of a random sample is a discrete distribution that gives weight $1/n$ to each point in a sample of size n.

The **empirical distribution function** or **EDF** is just the CDF of this empirical distribution.

If $\mathbb{I}(\cdot)$ is the **indicator function**, then the EDF is

$$\hat{F}(x) \equiv \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(x_i \leq x). \tag{3}$$

As we have seen, this is a step function.

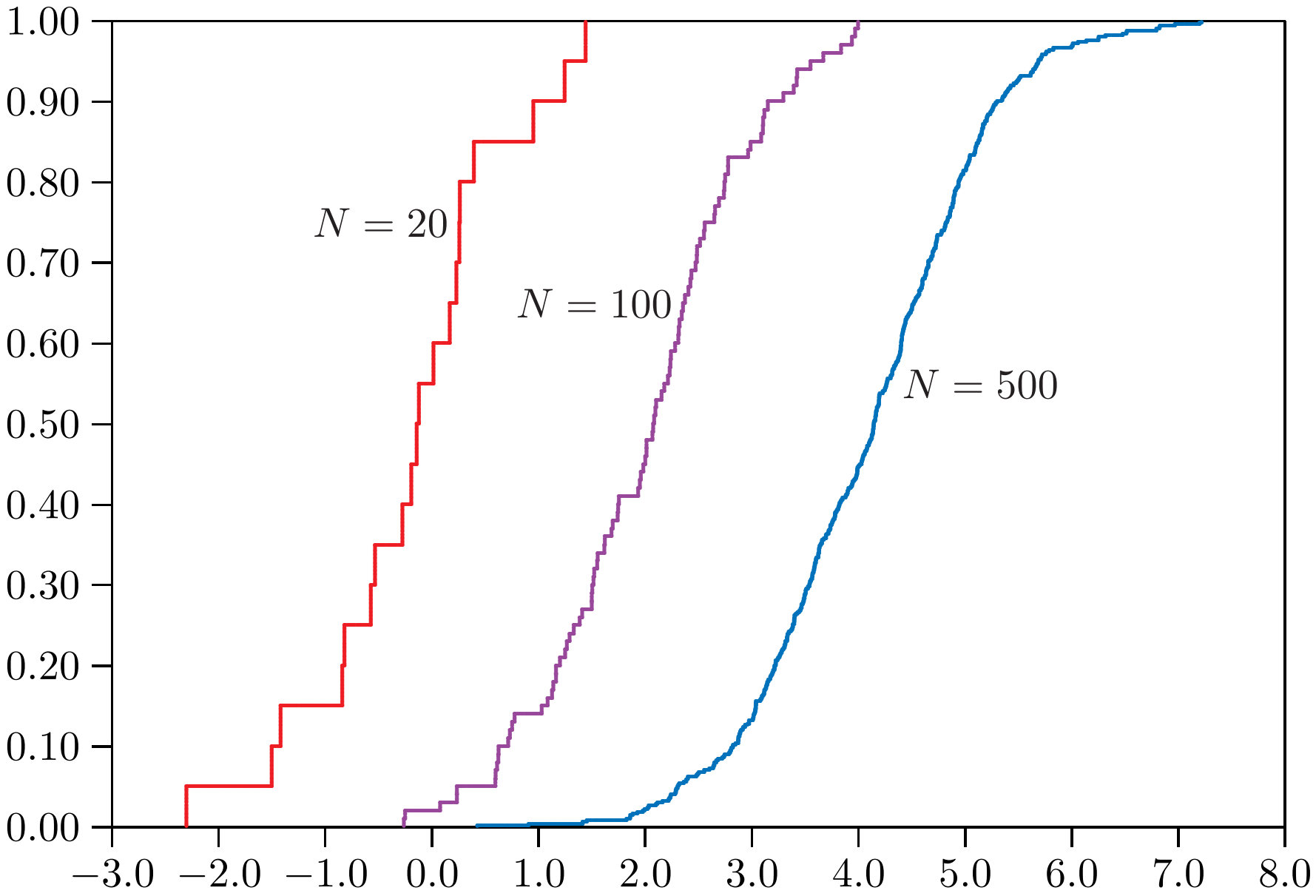
The height of each step is $1/n$, and the width is the difference between two successive values of x_i.

Figure 16.1 shows the EDFs for three samples of sizes 20, 100, and 500 drawn from three normal distributions, each with variance 1 and with means 0, 2, and 4 to avoid overlap.

Every bootstrap sample is a random drawing from the empirical distribution of the sample. That is what **resampling** amounts to.

For sufficiently large samples, EDF of the data should approximate the actual distribution of the data by the **fundamental theorem of statistics**. Thus $\hat{f}_{\text{bag}}(x_0) \cong \hat{f}_{\text{ag}}(x_0)$ should converge to $f(x_0)$.

Figure 16.1 — EDFs for three sample sizes



The expectation of $\hat{f}_{\text{bag}}(x_0)$ will differ from $\hat{f}(x)$ only when the latter is a nonlinear or adaptive function of the data. It would make no sense to bag linear regression predictions, or splines with fixed knots.

When $\hat{f}(x)$ is nonlinear in y (which it always is for trees), the bagging estimate may be substantially more efficient than $\hat{f}(x)$ itself.

In general, bagging lets us avoid finding a tuning parameter.

In the case of regression trees, bagging also lets us avoid pruning.

- Suppose we construct B regression trees, each using a different bootstrap training set. Then we average the resulting predictions.
- Because the trees are grown deep (not pruned), they tend to have high variance but low bias.
- The averaging reduces the variance without increasing bias, as pruning would do.
- It has been claimed that bagging is equivalent to pruning a latent "true" tree ("To bag is to prune," Coulombe, 2020).

For regression trees, it is obvious how to form a bagging prediction.

For classification trees, there is more than one possibility.

- The easiest approach is to find the **consensus** or majority prediction over the B trees.
- If there are 1000 trees and the most common prediction for x_0 is outcome 2, then we assign class 2 to x_0.
- A better approach is often to average the class probabilities across bootstrap trees.
- The probability assigned to class k for x_0 by any tree is just the proportion of observations in the node to which x_0 belongs, say node m, that are of class k. This is called $\hat{p}_{mk}$.
- So the bagged probability of class k is simply

$$\hat{p}_{mk}^{\text{bag}} = \frac{1}{B} \sum_{b=1}^{B} \hat{p}_{mk}^{b}. \tag{4}$$

  when x_0 belongs to node m.

In principle, bagging can be used with any nonlinear procedure for supervised learning.

In practice, it is especially popular for trees.

In order to implement bagging for trees, we can use the `randomForest` package. We will learn why shortly.

The default number of trees is 500. I used 1000.

I grew two sets of trees for the mileage data. The first one just had two predictors, `hp` and `wght`, as before.

The results were not great. The standard error of the predictions for the training sample was 0.00787, slightly worse than the value for OLS, which was 0.00717. But these may not be comparable. See Figure 16.2.

When I added four more predictors (number of cylinders, displacement, year, and origin), the fit improved a lot. The standard error was 0.00527, versus 0.00571 for OLS. See Figure 16.3.

Figure 16.2 — Fit vs. actual for bagged trees with two predictors

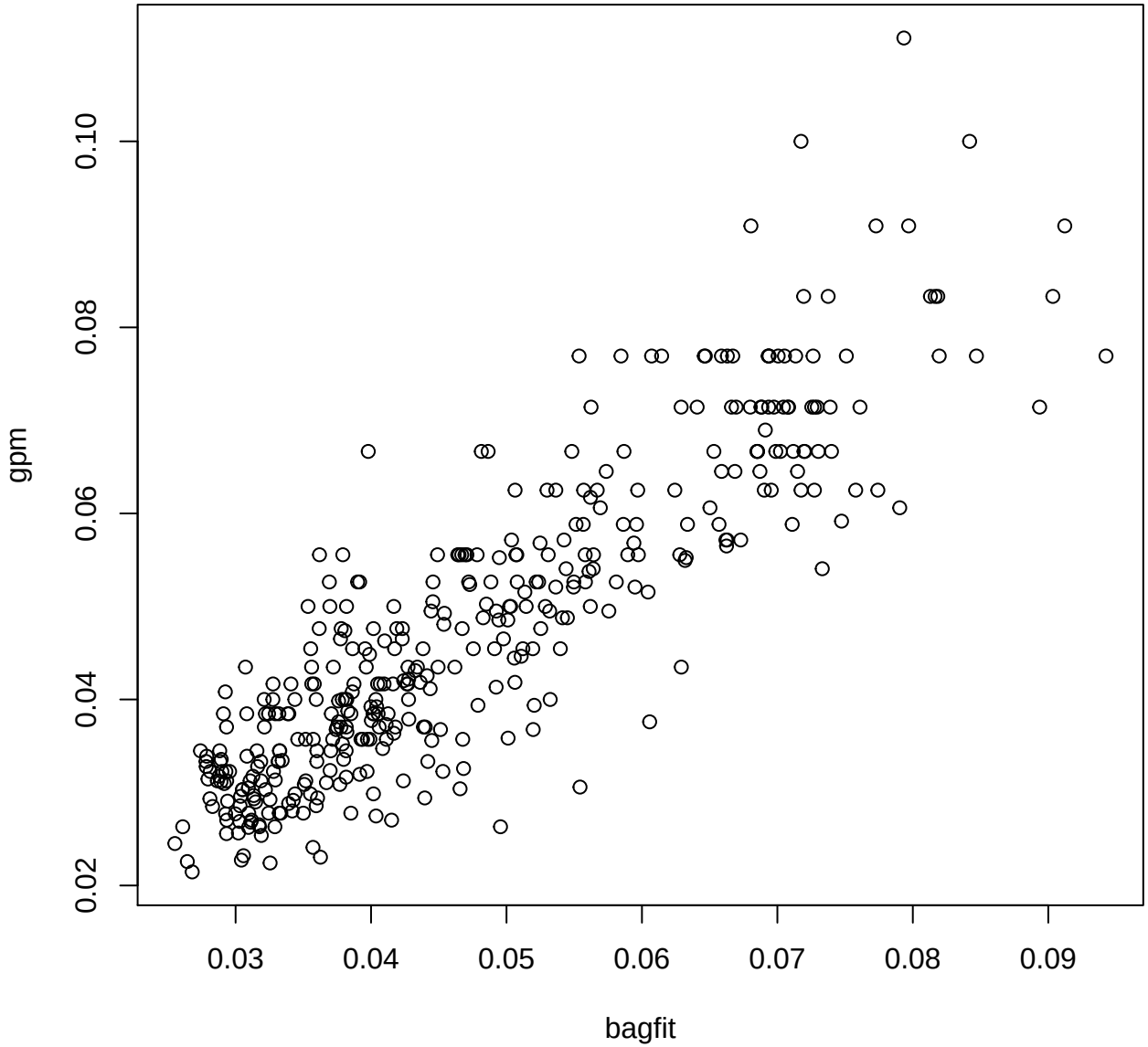
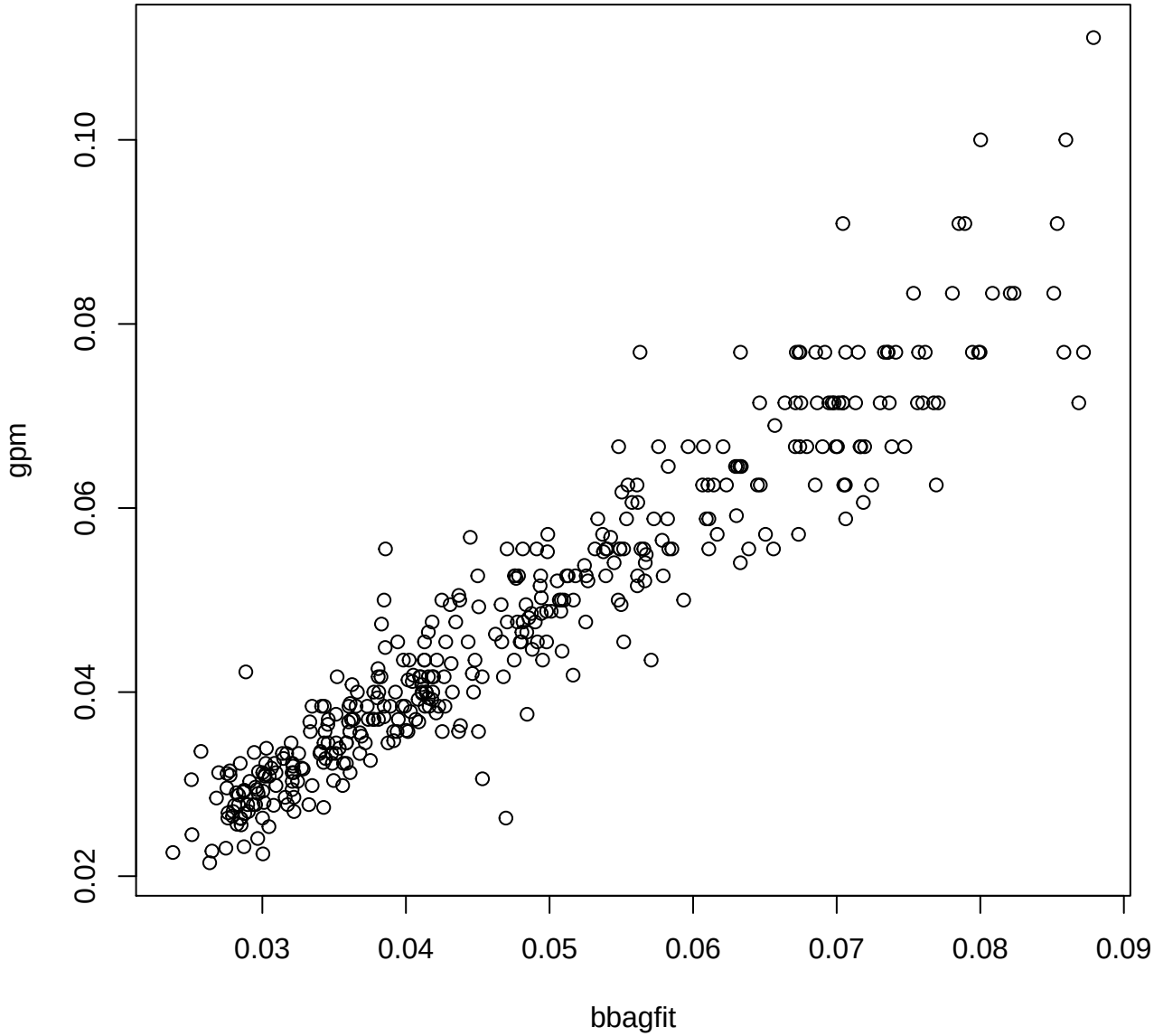


Figure 16.3 — Fit vs. actual for bagged trees with six predictors



The number of trees that we bag should be reasonably large; the fact that the default is 500 suggests that this must be the case.

Using just a handful of bootstrap samples will often produce results worse than the ones from a single tree.

This is illustrated in Figure 16.4, which shows two sequences of bagged trees, based on different seeds.

- In both cases, bagging based on $B < 8$ usually works worse than the original tree.
- But for $B \geq 8$, the bagged models always work better.

All four bagged models with $B = 640$ and $B = 1280$ yield essentially identical results, which are noticeably better than for the single tree.

Figure 16.4 is based on the training data, but similar results would be obtained for test data, if we had any; see Figure 16.5, from ESL.

Figure 16.5 also shows why using the consensus instead of the average probability can be a bad idea.

Figure 16.4 — Two sequences of bagged trees

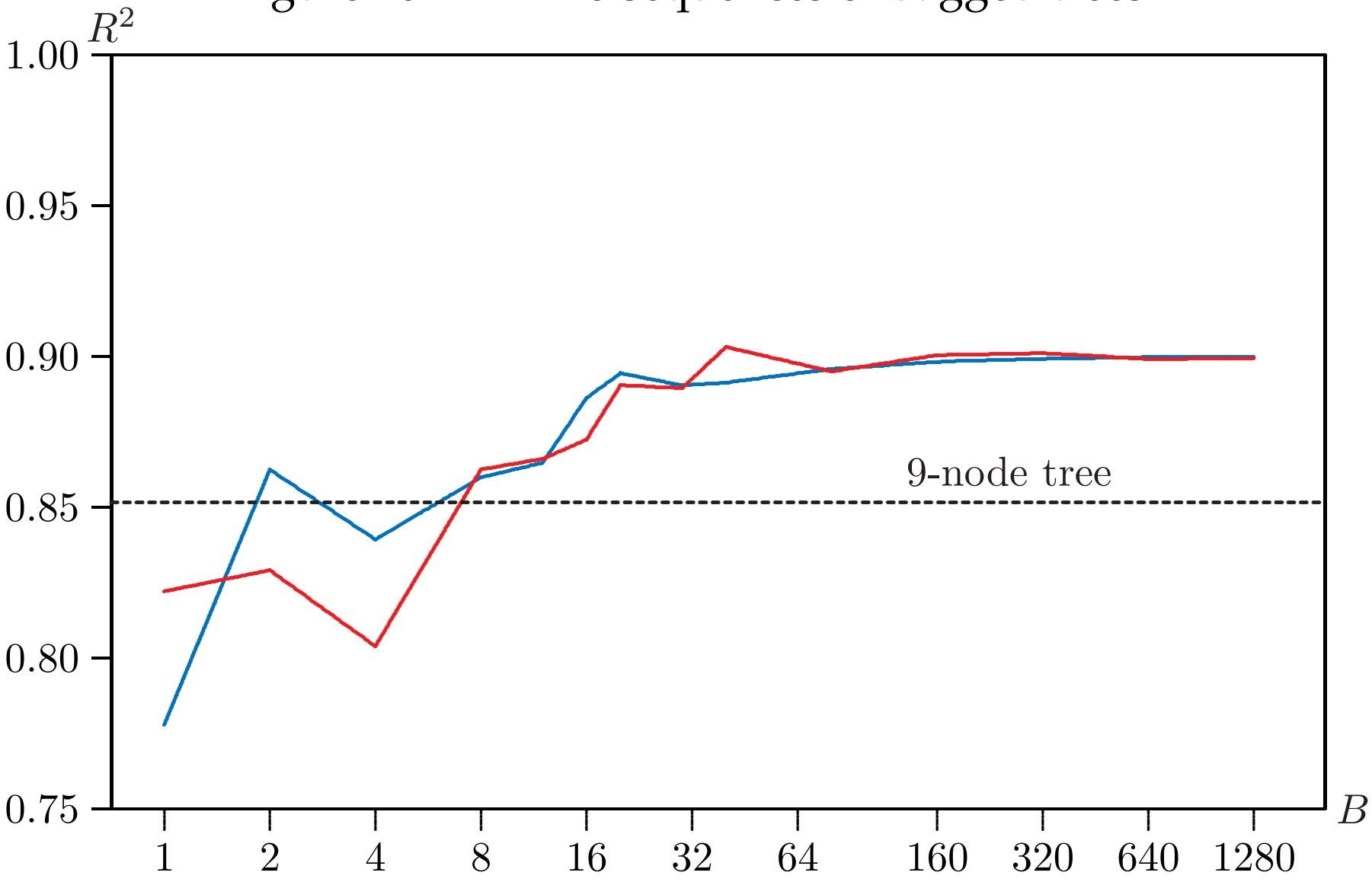


Figure 16.5 – Number of bagged trees versus test error

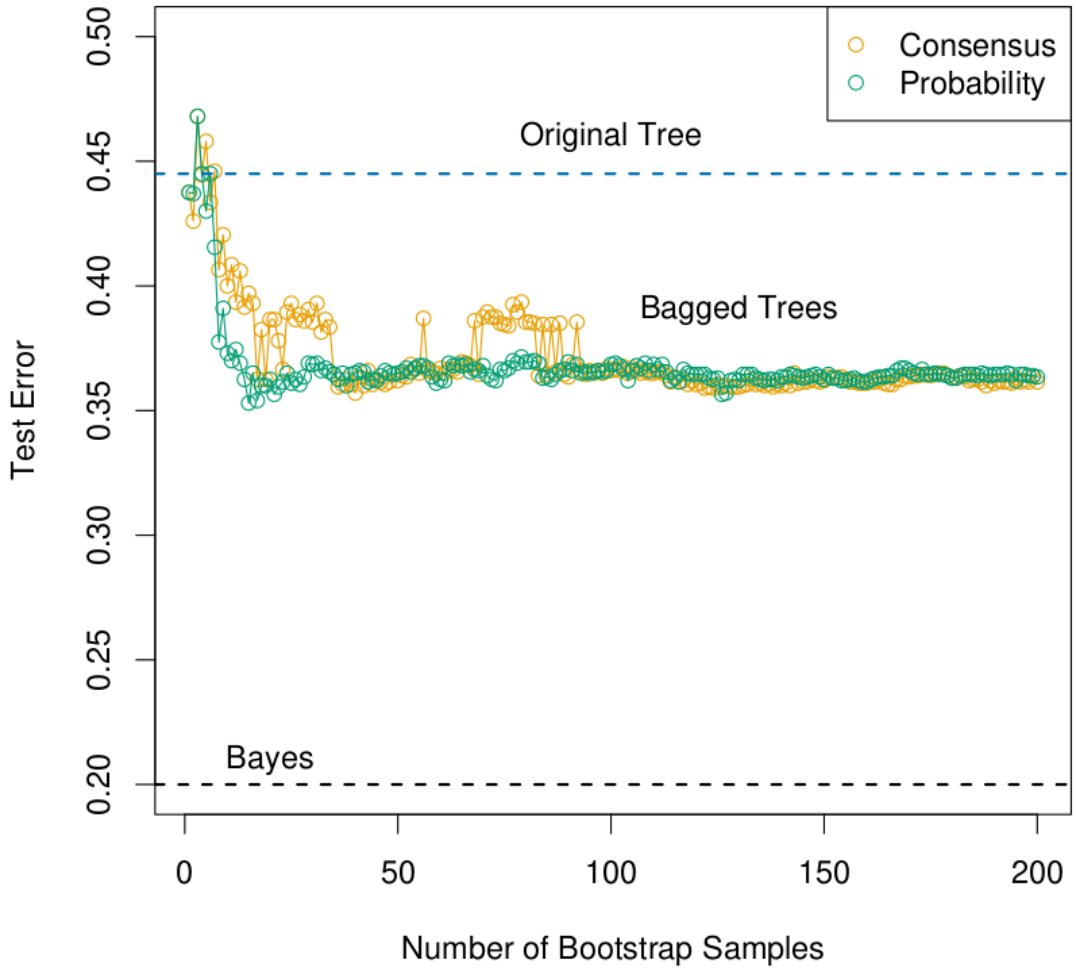


FIGURE 8.10. *Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The orange points correspond to the consensus vote, while the green points average the probabilities.*

Out-of-Bag Error Estimation

There is a remarkably simple way to estimate the test error of any bagged model.

There is no need to obtain a validation set, and there is no need to perform explicit cross-validation.

- The key to bagging is that each tree is grown using a different bootstrap sample.
- On average, as we saw in Slides 8, each bootstrap sample omits about 36.8% of the observations.
- The observations omitted from any bootstrap sample are called **out-of-bag**, or **OOB**, observations.
- For each observation i, we can find all the bootstrap samples that omitted it. Then we average their predictions.
- This is the OOB prediction for observation i. Call it $\hat{y}_i^{\text{oob}}$.

The OOB measure of mean squared error is simply

$$\text{MSE}^{\text{oob}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i^{\text{oob}})^2. \tag{5}$$

This tends to be very similar to what we would obtain from n-fold cross-validation.

In effect, we can bag and cross-validate at the same time. As we obtain more bootstrap samples, MSE^{oob} should settle down, and we can stop once it has done so.

One problem with bagged trees is that they are much harder to understand than single trees.

However, it is possible to measure the `importance` of each predictor. This is very useful.

For bagged regression trees, the importance of an input x can be measured in several ways.

The most common method is to measure the improvement in fit from splitting on each variable, averaged over all trees.

If the fit improves a lot, on average, then a predictor is important.

- For regression, the fit is measured by the RSS.
- For classification, the fit is measured by the Gini index, or perhaps some other measure of node impurity.

The left panel of Figure 16.6 shows one measure, and the right panel shows the second. Dropping `cyl` and `origin` does not help.

Figure 16.7, which is Figure 8.9 in ISLR/ISLP, measures variable importance for the heart disease data using the Gini index.

The values in Figure 16.7 seem to be normalized so that the largest one equals 100.

Figure 16.6 — Variable Importance Plots for Bagged Trees

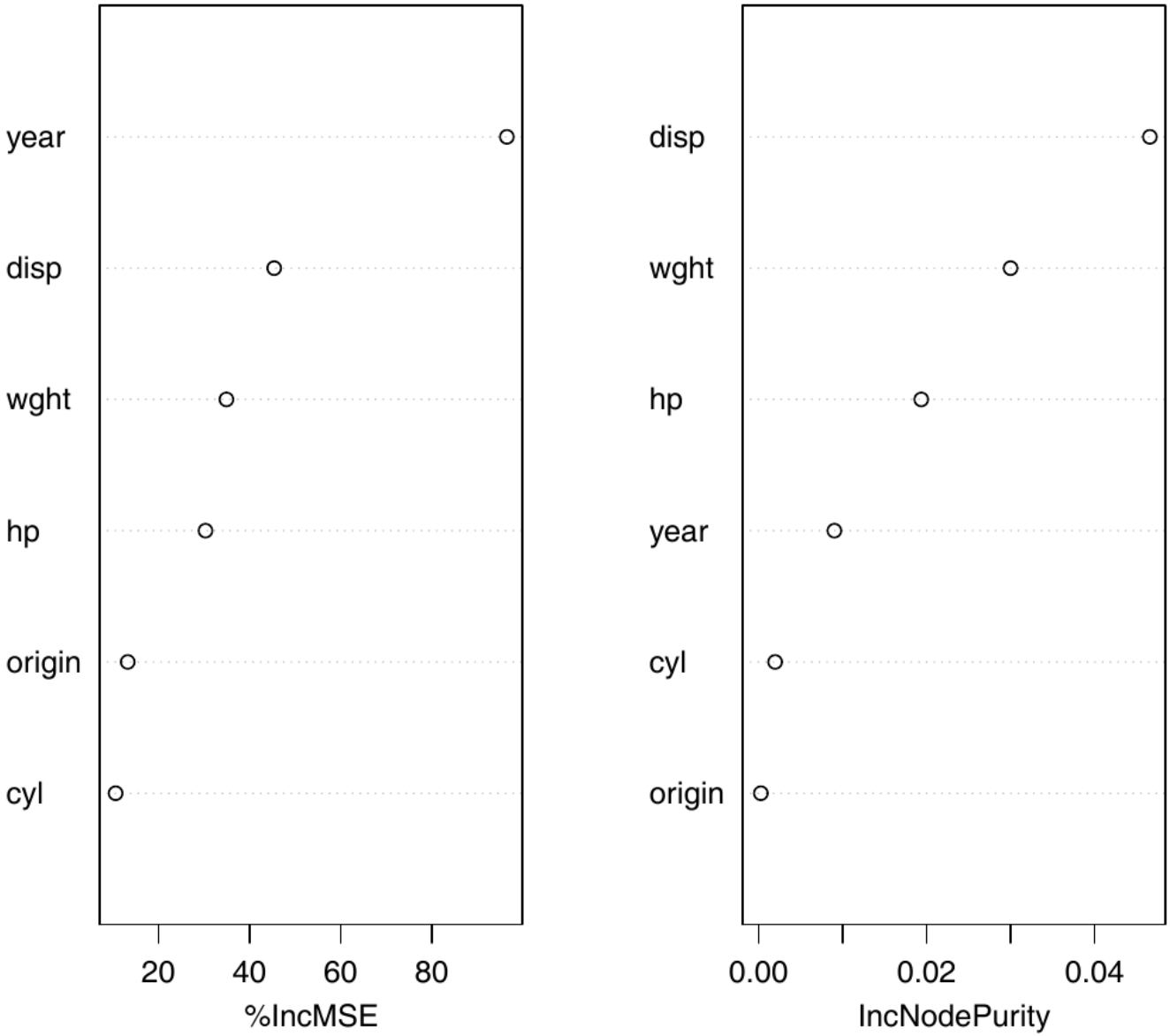
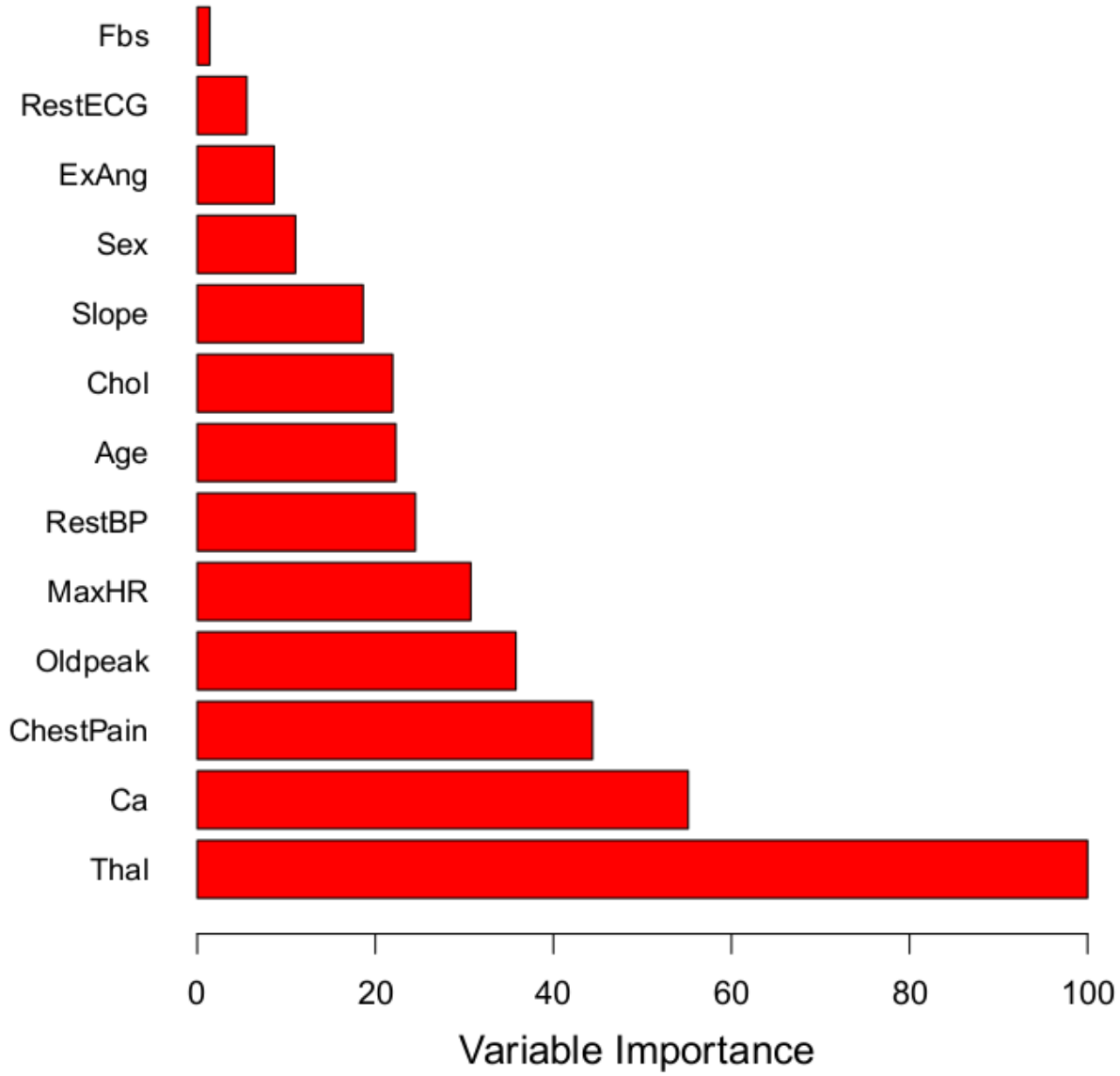


Figure 16.7 — Another Variable Importance Plot



Random Forests

Random forests (Breiman, 2001) is a widely-used method that is easy to use and often works well.

Interestingly, Breiman was 72 or 73 when this paper appeared, and it has over 146,292 citations. Unfortunately, he died four years later.

The `randomForest` package in R is based on Fortran code written by Breiman and Cutler.

As we just saw, bagging (that is, averaging over the predictions from a number of bootstrap samples) reduces variance but not bias.

This can work well if all the models are approximately unbiased and not very correlated with each other.

If we average B random variables y_b each with variance σ^2, the variance of the average is σ^2 / B.

If the random variables are correlated, with variance σ^2 and covariance $\rho\sigma^2$, we instead find that

$$\text{Var}(\bar{y}) = \frac{1}{B^2}\sum_{b=1}^{B}\text{Var}(y_b) + \frac{2}{B^2}\sum_{b=1}^{B}\sum_{b'=b+1}^{B}\text{Cov}(y_b, y_{b'}) \tag{6}$$

$$= \frac{1}{B}\sigma^2 + \frac{1}{B^2}B(B-1)\rho\sigma^2 \tag{7}$$

$$= \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2. \tag{8}$$

As B increases, the second term tends to zero, like σ^2/B in the uncorrelated case.

But the first term does not tend to zero. No matter how many random variables we average (in this case, predictions from different models), we cannot reduce the variance below $\rho\sigma^2$.

This suggests that we can do better than bagging if we can reduce the correlation among the trees.

- The trick is not to allow all possible splits. Instead, the algorithm randomly selects m out of p possible variables for splitting.
- It does this at each split. Thus, each tree may ultimately involve splits on every input. But the order of the splits is likely to be very different across trees.

Typically, m is quite small, like $\sqrt{p}$. That is the default for classification. The default for regression is $p/3$.

If $m = p$, random forests is simply bagging applied to trees. That is why I used the `randomForest` function for bagging above.

By not allowing all possible splits, we reduce the correlations among the trees.

Suppose there is one very strong predictor. Then all or most bagged trees will normally split on it first. Thus they will resemble each other.

But if we do not allow 2/3 of the trees to split initially on the strong predictor, they will look much less like each other.

Using a small value of m can be particularly helpful when the inputs are highly correlated.

For the mileage dataset, however, using random forests with $m < p$ worked no better than bagging.

In fact, using the same seed for each random forest, R^2 increased monotonically with m:

$$
\begin{aligned}
m = 1 &\longrightarrow 0.8681 \quad \text{(bad choice)} \\
m = 2 &\longrightarrow 0.8932 \quad (p/3) \\
m = 3 &\longrightarrow 0.8981 \\
m = 4 &\longrightarrow 0.8996 \\
m = 5 &\longrightarrow 0.9000 \\
m = 6 &\longrightarrow 0.9004
\end{aligned}
$$

The bias of a random forest is the average bias of the trees.

- Like pruning, reducing m tends to increase bias and reduce variance.
- As discussed for bagging, it is not really necessary to prune trees that belong to random forests, although we should specify the minimum node size.
- Most importantly, we have to choose m, and the default values may not work well.

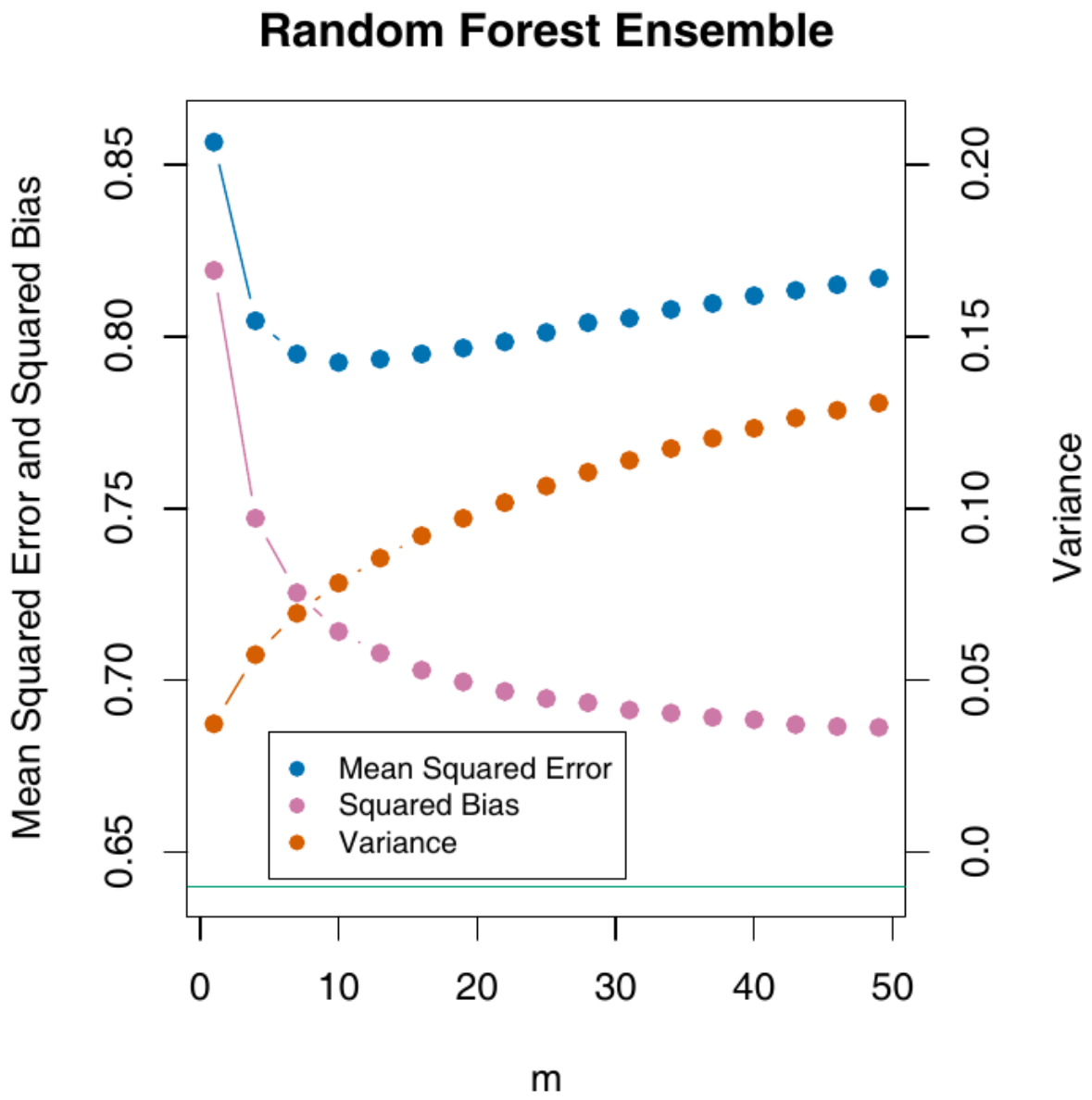
Figure 16.8 (from ESL, Chapter 15) illustrates the tradeoff for m. This is for a case with 50 inputs and an extremely symmetric additive model.

The optimal value of m seems to be about 10.

Making m larger than that continues to reduce bias, but at the expense of increased variance.

As with bagging, of which random forests are just a special case, we must also choose B, but here larger is always better.

Figure 16.8 (from ESL)



- It is not clear what we learn from a case with additivity, where an additive method like OLS or ridge would probably work better.
- In fact, ESL admits that ridge works considerably better than random forests here, with MSE about 0.45.
- ESL also points out that random forests with m small can behave a lot like ridge!
- When m is small, the relevant variables take turns being the primary split. In the case of Figure 16.8, every variable is equally relevant.
- The ensemble averaging then reduces the contribution of any individual variable.

Like ridge and lasso, random forests can be used quite mechanically even when p is large.

A recently developed method called **generalized random forests** goes beyond the classic random forests method in several respects.

The associated package, called `grf`, can be used instead of `randomForest`.