

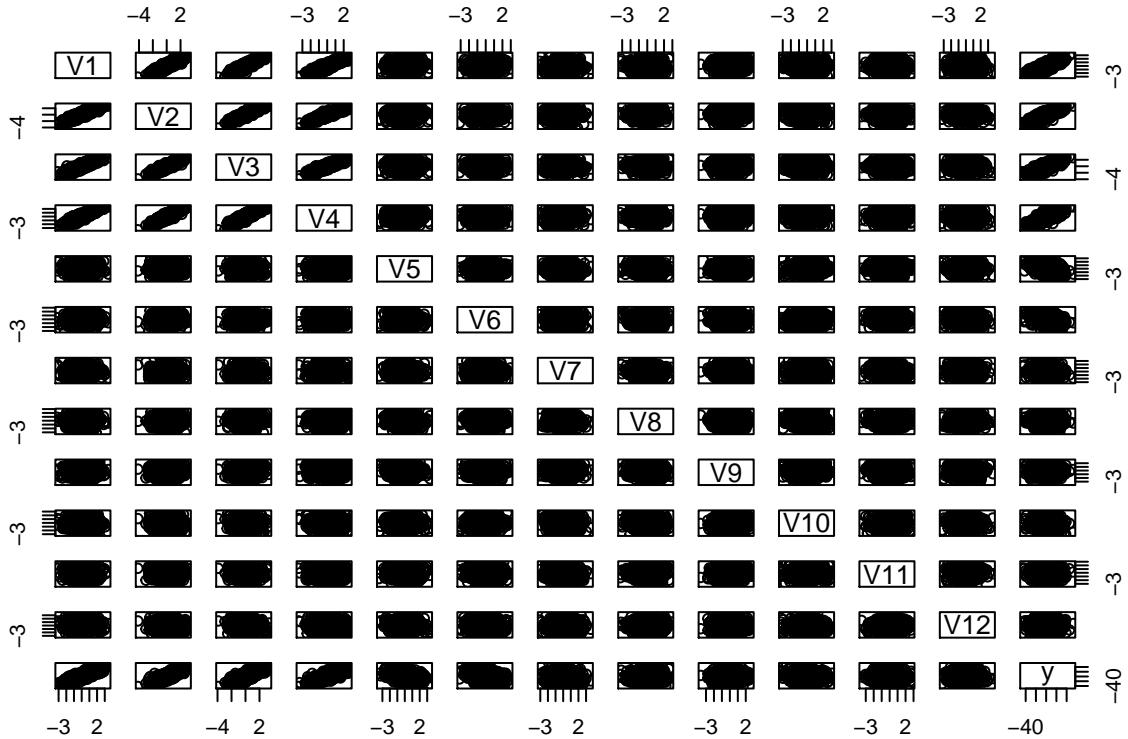
Simulations and Comparisons

Simulated Data

To aid in comparisons between the methods, one of the simulated datasets considered in this paper will be generated from the same method as used in (Strobl et al, 2008??). Under this method, the 13×1000 data set, D_1 , has 12 predictors, V_1, \dots, V_{12} , where $V_j \sim N(0, 1)$. The first four are, however, block correlated to each other with $\rho = .9$. They are related to Y by the linear equation:

$$Y = 5 \cdot V_1 + 5 \cdot V_2 + 2 \cdot V_3 + 0 \cdot V_4 + -5 \cdot V_5 + -5 \cdot V_6 + 0 \cdot V_7 + 0 \cdot \dots + E, E \sim N(0, \frac{1}{2})$$

Note that the coefficients for V_7, \dots, V_{12} are all zero.



##	V1	V2	V3	V4	V5
## V1	1.000000000	0.9146541132	0.907895351	0.907066765	-0.034234292
## V2	0.914654113	1.000000000	0.913780977	0.913731929	-0.020066937
## V3	0.907895351	0.9137809775	1.000000000	0.902766836	-0.016703104
## V4	0.907066765	0.9137319288	0.902766836	1.000000000	-0.002169185
## V5	-0.034234292	-0.0200669374	-0.016703104	-0.002169185	1.000000000
## V6	0.006059950	-0.0007016153	-0.006844401	-0.014814462	0.043610533
## V7	0.011864999	-0.0013307105	0.006951044	0.022724703	0.005306628
## V8	0.012429245	0.0180018750	0.021232744	0.020871464	-0.031895007
## V9	0.035378928	0.0383359616	0.036338439	0.037906433	-0.024882964
## V10	-0.027164872	-0.0205322792	-0.021277153	-0.021746391	0.005239249
## V11	-0.062839803	-0.0447873818	-0.029615955	-0.053826143	-0.018439527
## V12	0.001895378	-0.0040846930	-0.005222882	-0.017304444	-0.043550921
## y	0.828867870	0.8301277767	0.808338105	0.788791607	-0.388422174
##	V6	V7	V8	V9	V10
## V1	0.0060599505	0.011864999	0.01242924	0.035378928	-0.027164872
## V2	-0.0007016153	-0.001330711	0.01800187	0.038335962	-0.020532279

```

## V3 -0.0068444008 0.006951044 0.02123274 0.036338439 -0.021277153
## V4 -0.0148144624 0.022724703 0.02087146 0.037906433 -0.021746391
## V5 0.0436105331 0.005306628 -0.03189501 -0.024882964 0.005239249
## V6 1.0000000000 -0.004856916 -0.02539841 -0.013632377 0.036331634
## V7 -0.0048569164 1.000000000 -0.02367510 0.006567436 0.018156112
## V8 -0.0253984113 -0.023675099 1.00000000 -0.038840769 -0.089374372
## V9 -0.0136323768 0.006567436 -0.03884077 1.000000000 0.001732069
## V10 0.0363316342 0.018156112 -0.08937437 0.001732069 1.000000000
## V11 -0.0217988171 -0.033514884 0.02826398 -0.075004886 0.010676447
## V12 -0.0422332641 0.032644574 0.02539601 0.009560373 0.015536556
## y -0.3637688830 -0.141169908 0.03665375 0.045997408 -0.037738613
##           V11          V12          y
## V1 -0.06283980 0.001895378 0.82886787
## V2 -0.04478738 -0.004084693 0.83012778
## V3 -0.02961595 -0.005222882 0.80833811
## V4 -0.05382614 -0.017304444 0.78879161
## V5 -0.01843953 -0.043550921 -0.38842217
## V6 -0.02179882 -0.042233264 -0.36376888
## V7 -0.03351488 0.032644574 -0.14116991
## V8 0.02826398 0.025396015 0.03665375
## V9 -0.07500489 0.009560373 0.04599741
## V10 0.01067645 0.015536556 -0.03773861
## V11 1.00000000 0.022521656 -0.02183787
## V12 0.02252166 1.000000000 0.02427605
## y -0.02183787 0.024276053 1.000000000

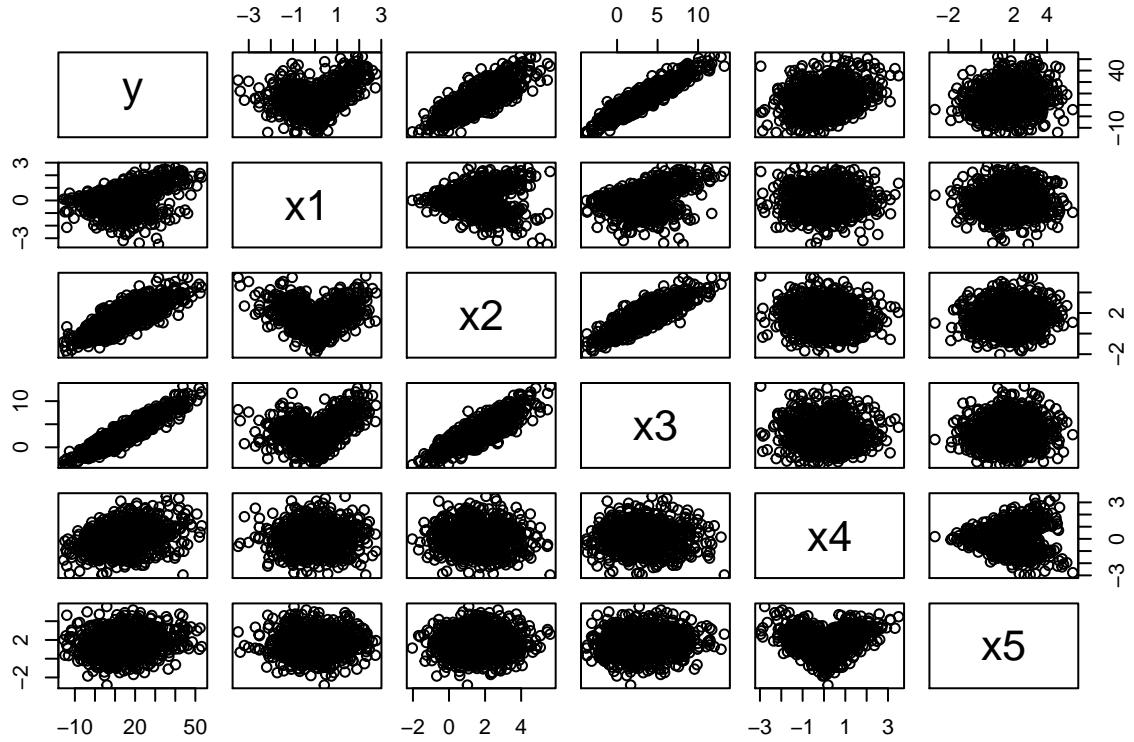
```

Let's move on to a more difficult situation. The dataset $D2$ contains five predictors, X_1, \dots, X_5 , that have an interesting structure- several of the predictors are correlated, but are not one -to -one. This violates an important assumption of the linear model and means that these variables have low correlation. Note that this only makes sense in higher dimensions where we are estimating the value of X_j given X_1, \dots, X_n .

```

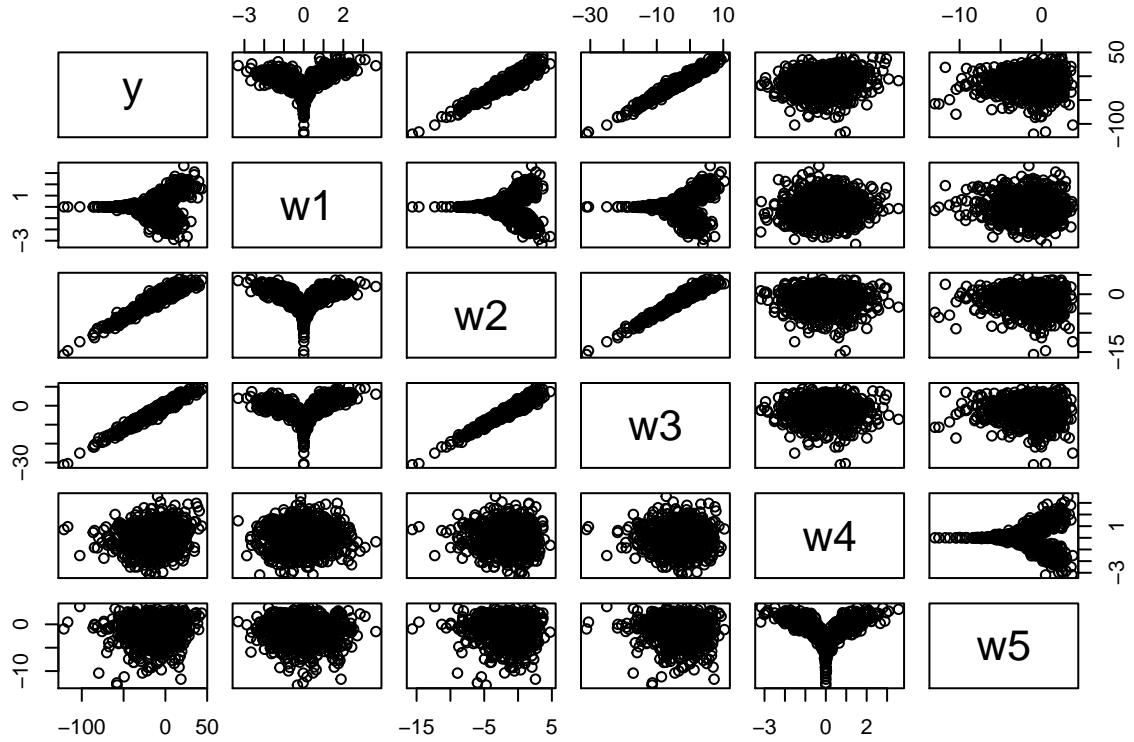
##           y          x1          x2          x3          x4          x5
## y 1.0000000 0.37370066 0.81852208 0.92926643 0.30615522 0.15357237
## x1 0.3737007 1.00000000 0.01253512 0.37828457 0.02553116 0.01196494
## x2 0.8185221 0.01253512 1.00000000 0.85944303 -0.04770287 0.04253338
## x3 0.9292664 0.37828457 0.85944303 1.00000000 -0.03116991 0.04579883
## x4 0.3061552 0.02553116 -0.04770287 -0.03116991 1.00000000 0.01984835
## x5 0.1535724 0.01196494 0.04253338 0.04579883 0.01984835 1.00000000

```



The trickier relationship between the variables in $D2$ was because they were generated using the square root of the absolute value of the other variable. Here, in $D3$ the process is repeated but with the log of the absolute value.

```
##          y      w1      w2      w3      w4      w5
## y 1.0000000 0.15728407 0.94764704 0.97416878 0.17279746 0.05578763
## w1 0.15728407 1.00000000 -0.02714818 0.15752508 0.03050113 -0.05109218
## w2 0.94764704 -0.02714818 1.00000000 0.96531545 -0.01975103 -0.04414540
## w3 0.97416878 0.15752508 0.96531545 1.00000000 -0.01042494 -0.05419898
## w4 0.17279746 0.03050113 -0.01975103 -0.01042494 1.00000000 -0.03325170
## w5 0.05578763 -0.05109218 -0.04414540 -0.05419898 -0.03325170 1.00000000
```



Models and Comparisons

Trees

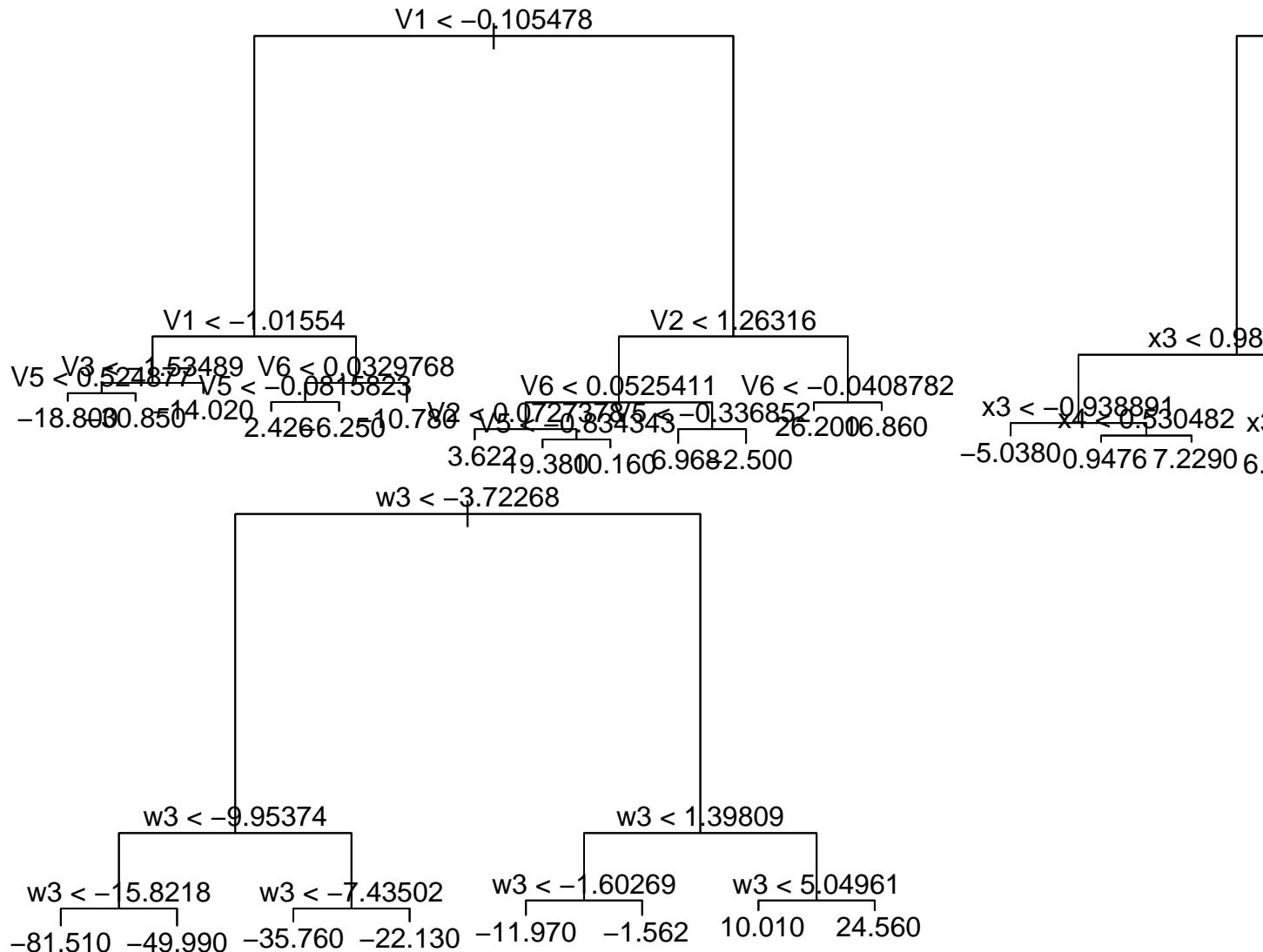
CART: Regression Trees

As outlined in the 1984 textbook, *Classification and Regression Trees*, Breiman, Friedman, Olshen, and Stone described their method for creating, pruning, and testing regression trees. What follows is their basic algorithm for fitting regression trees:

Algorithm 2: CART, Regression Trees

- 1.
- 2.
- 3.
- 4.

Let's return to our simulated datasets. Here is what trees grown for each dataset using the model $Y \sim X_1, \dots, X_n$.



Conditional Inference Trees

Another popular method for creating regression tree models is Conditional Inference Trees.

CI trees

1. For case weights w test the global null hypothesis of independence between any of the m covariates.
 2. Choose a set $A \subset \{j\}$ in order to split X_j into two disjoint sets A and $\{j\} \setminus A$.
 3. Recursively repeat steps 1 and 2 with modified case weights w_{left} and w_{right} , respectively.
- from <https://eeecon.uibk.ac.at/~zeileis/papers/Hothorn+Hornik+Zeileis-2006.pdf>

After step 1 is completed, any goodness of fit method can be used to generate the split and choose the set A . Note that in this method the splitting is done separately from the variable selection.

Bagged Forests

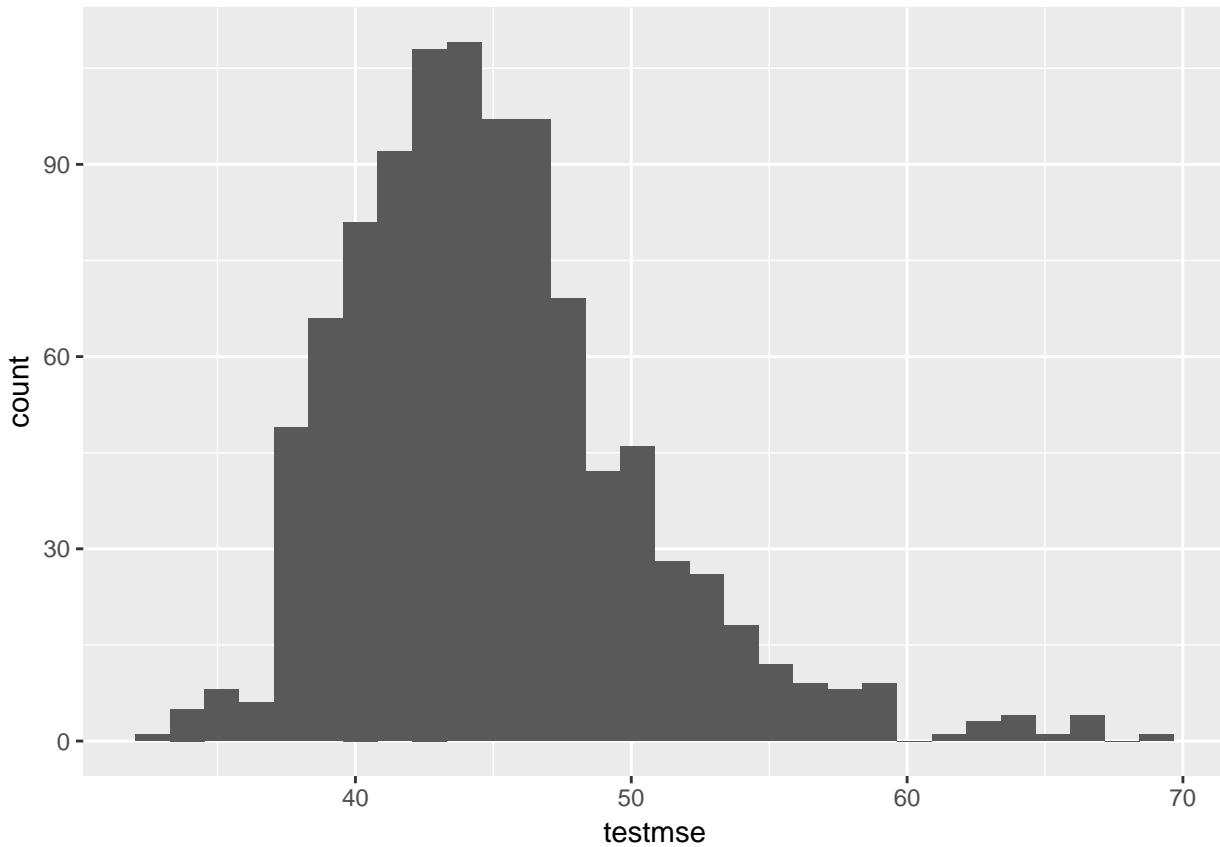
As mentioned in the introduction, single trees can have some variability that in practice, limits their use. This simulation demonstrates this principle:

```
for 1000 trials:
```

1. separate D3 into a training set and a test set where the training set contains 2/3 of the observations
2. fit a CART tree
3. Predict the response using the test set
4. Calculate the mean squared error

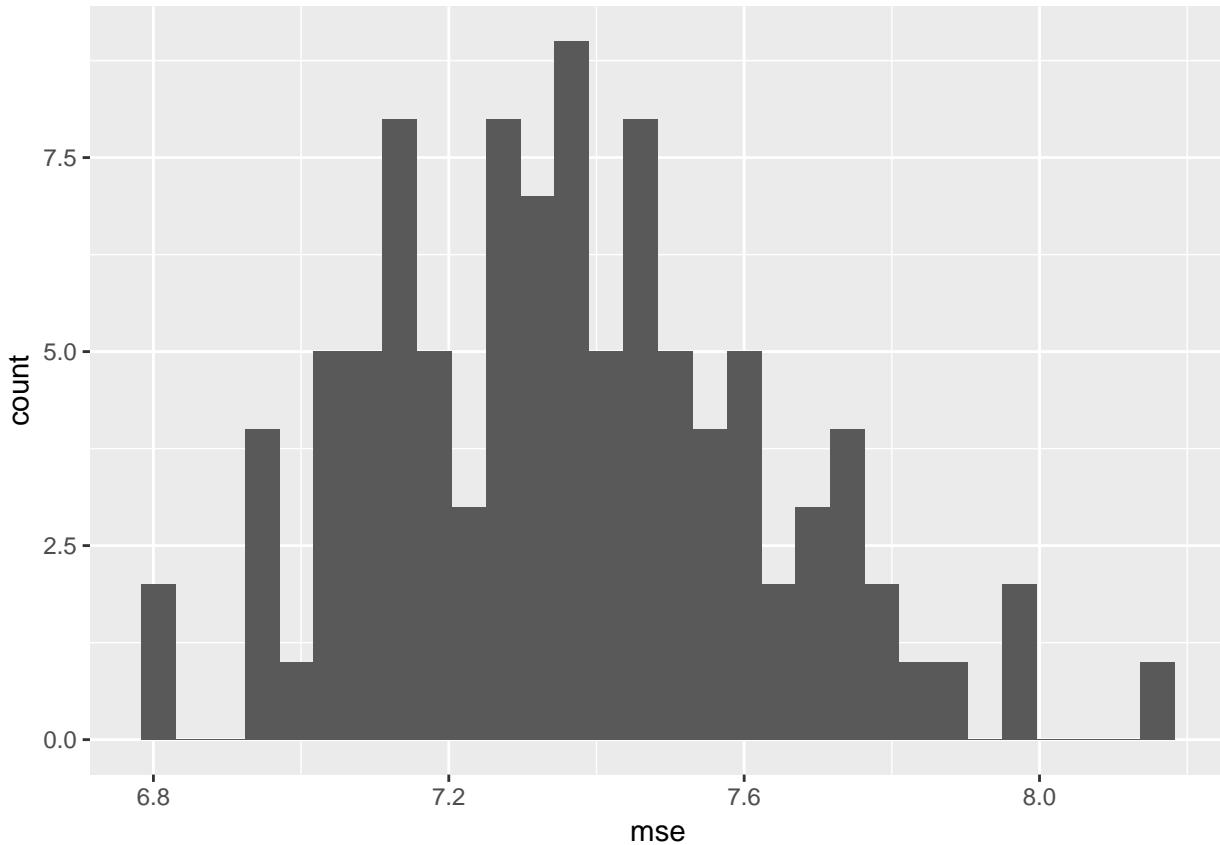
Here is a histogram of the MSE found using this method:

```
##          testmse
## testmse 28.56955
```



As one can see, there is a fair amount of variability in a single tree, they are heavily dependent on fluctuations in the starting data set. As mentioned briefly in the introduction, bagged forests present one solution to this problem. To create a bagged forest, as outlined in *An Introduction to Statistical Learning* by James, Witten, Hastie and Tibshirani, 2013, many bootstrap samples are taken from the initial dataset and trees are fitted to them. The final predictions are, then, averaged over all of the trees. This ensures that while each tree has high variance, when they are aggregated the variance will decrease.

Let's put that to the test here using our dataset *D3* again. We'll build 100 forests of 100 trees each and compare the variability of the *MSE* distributions.



```
##           mse
## mse 0.07051985
```

As one can see, the values of MSE_{test} for the bagged forest were entirely below the MSE_{test} for the trees and the variance was much smaller.

Random Forests

As the number of trees grown in each forest increases, the MSE_{test} decreases (cite). Still, this can become computationally intensive on larger data sets where we would like very accurate models. Random forests are often seen as a solution to this problem. In a bagged forest, every variable is considered when each split is made but in a random forest only $mtry, mtry \leq p$ are considered. This allows us to assume that the trees have a level of independence not found in bagged forests, and that a small random forest will often out perform the bagged forest.

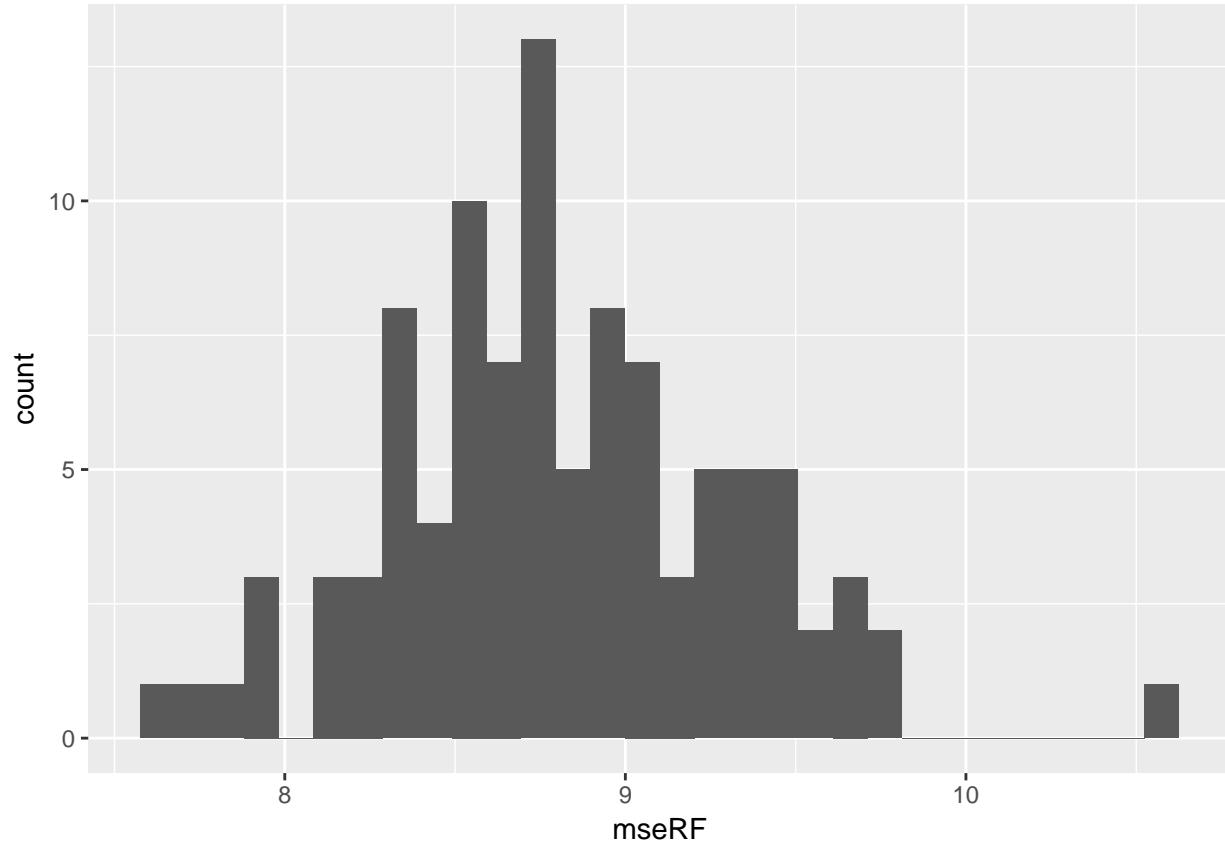
For an illustration, let's build a random forest on $D3$ and compare the MSE .

(include note about oob error vs test mse)

```
mseRF <- rep(0,100)
train = sample(1000,666)
for (i in 1:100){
  rf <- randomForest(y~., data = d3, subset = train, ntree = 100)
  mseRF[i] <- mean((d3$y[-train] - predict(rf, d3[-train,]))^2)
}

mseRF <- as.data.frame(mseRF)
```

```
ggplot(aes(x = mseRF), data = mseRF) + geom_histogram()  
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
var(mseRF)  
##          mseRF  
## mseRF 0.2528736
```