Chapter 8: Tree Based Methods

- Tree based methods can be used for regression or classification.
- These methods depend on dividing the predictor space into regions following a decision tree.
- Basic trees are used as building blocks for more powerful methods like bagging, random forests, and boosting.

Regression Trees

- The regression trees creates nodes which satisfy a single condition, e.g. $X_j < t_k$.
- The terminal nodes of the tree are called leaves.
- In the next few examples the salary of major league baseball players is predicted based on a number of predictor variables for each player, like # of hits, years in the league, # of walks and so on.

Regression Trees

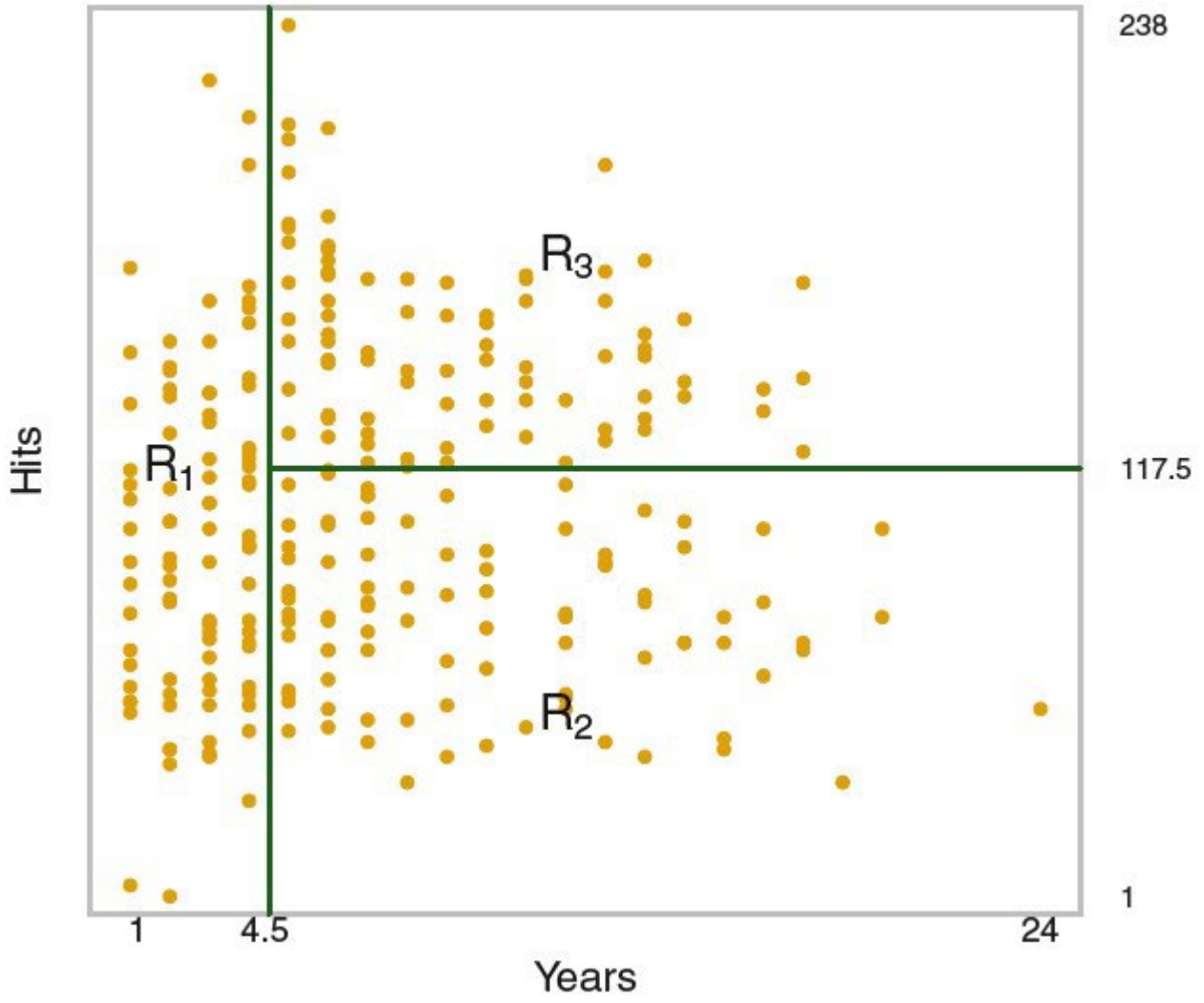
This tree can be interpreted as indicating that years is the most important factor in determining salary.

For new players hits has little impact on their salary but for more experienced players the number of hits becomes important.

FIGURE 8.1. *For the* `Hitters` *data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year. At a given internal node, the label (of the form* $X_j < t_k$*) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to* $X_j \geq t_k$*. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to* `Years<4.5`*, and the right-hand branch corresponds to* `Years>=4.5`*. The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.*

Regression Trees

The previous regression tree with three terminal nodes divides the predictor space of Hits and Years into three regions



Regression Trees

- Building the regression tree involves dividing the predictor space into J, distinct non-overlapping regions, R_1, …, R_j.
- Every observation in a region R_j, is assigned the mean of all the training observations that fell in the region.
- More formally the goal is to find regions that minimize the RSS of $\sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j} \right)^2$
- We can't look at all possible regions.
- Use recursive binary splitting.
- Starting at the top of the tree each split is based on the greatest reduction in RSS at that step, without looking ahead.

Regression Trees: recursive binary splitting

- For instance, at the first node we look at cutpoint, *s*, for predictor-*j* that divide the observations into two regions, $\{X|X_j<s\}$ and $\{X|X_j \geq s$
- Then we seek values of *j* and *s* that minimize,
  $\sum_{i:\, x_i \in R_1(j,s)} \left(y_i - \hat{y}_{R_1}\right)^2 + \sum_{i:\, x_i \in R_2(j,s)} \left(y_i - \hat{y}_{R_2}\right)^2$
  where $\hat{y}_{R_1}$ is the mean of the training observations in region $R_1(j,s)$ and $\hat{y}_{R_2}$ is the mean of the training observations in region $R_2(j,s)$.
- Next the process looks for splits at each of the two newly created regions choosing the best as before.
- This continues until some stopping criteria is reached such as no more than 5 observations in any region. Call this tree T_0.

Regression Trees: pruning

- The previously described build up of a regression tree is likely to overfit the data.
- Requiring each reduction in RSS to exceed a threshold is likely to prematurely stop a tree.
- Use cost complexity pruning (weakest link pruning).
- Look at subtrees, $T \subset T_0$, that by collapsing internal nodes. These subtrees are evaluated with a penalty function.
- Let the parameter α vary and find the subtree, T_α, that minimizes,
  $C_\alpha(T) = \sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2 + \alpha|T|$
  at each value of α, where $|T|$ indicates the number of terminal nodes of the tree T.

Regression Trees: pruning

- To find T_α successively collapse internal nodes starting using the node that produces the smallest per-node increase in $C_\alpha(T) - \alpha|T|$.
- This continues until there is only one node.
- This sequence of subtrees will contain T_α.
- When α=0 we just have T_0. As α gets larger the best tree will be smaller.
- Use cross-validation to pick the best α.

Building a Regression Tree

- Algorithm

  1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
  2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
  3. Use *K*-fold cross-validation to choose α. That is, divide the training observations into *K* folds. For each $k = 1,..,K$:

  (a) Repeat steps 1 and 2 on all but the kth fold of the training data.

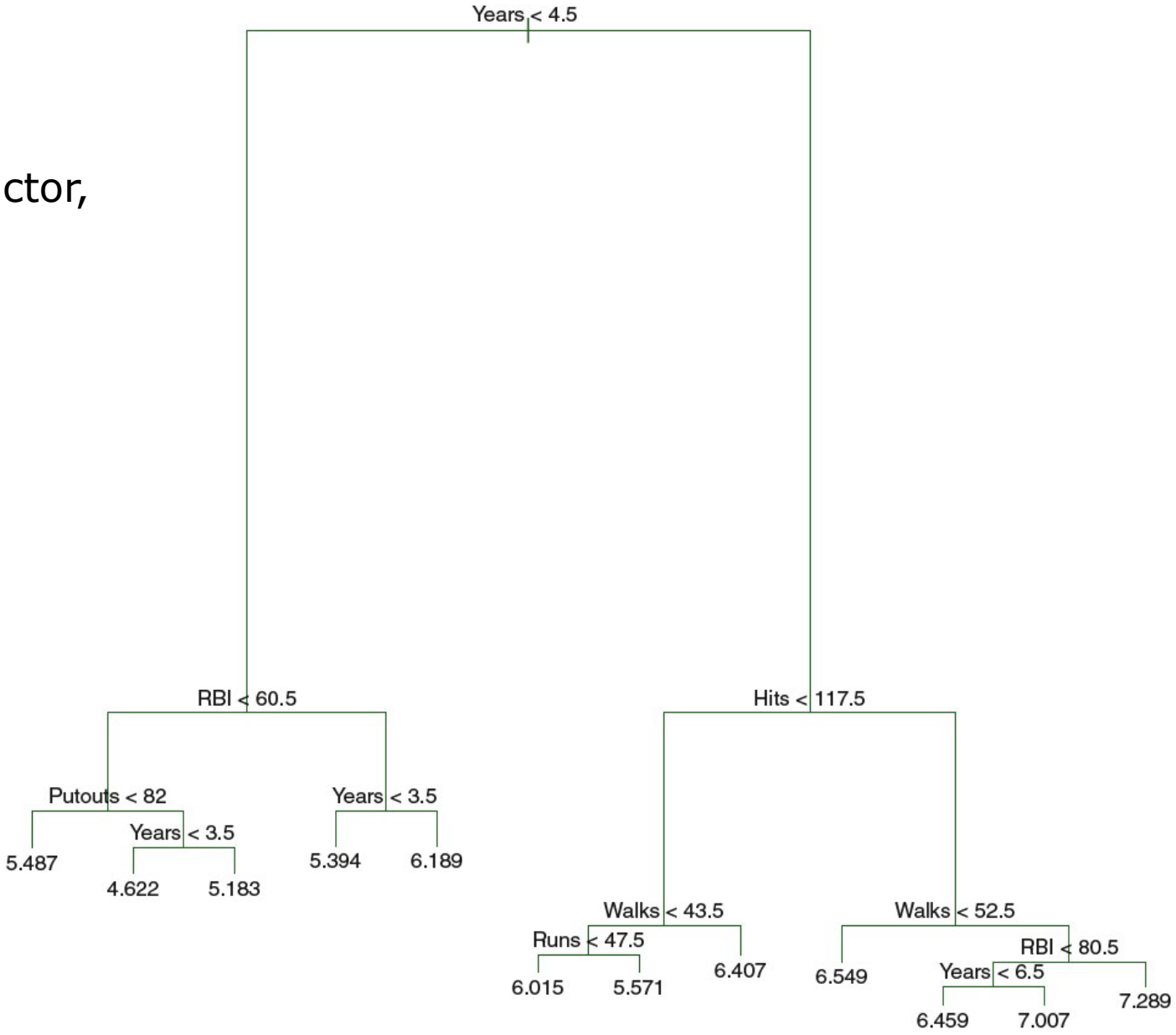
  (b) Evaluate the mean squared prediction error on the data in the left-out *kth* fold, as a function of α.

  Average the results for each value of α and pick α to minimize the average error.

  (4) Return the subtree from step 2 that corresponds to the chosen value of α.

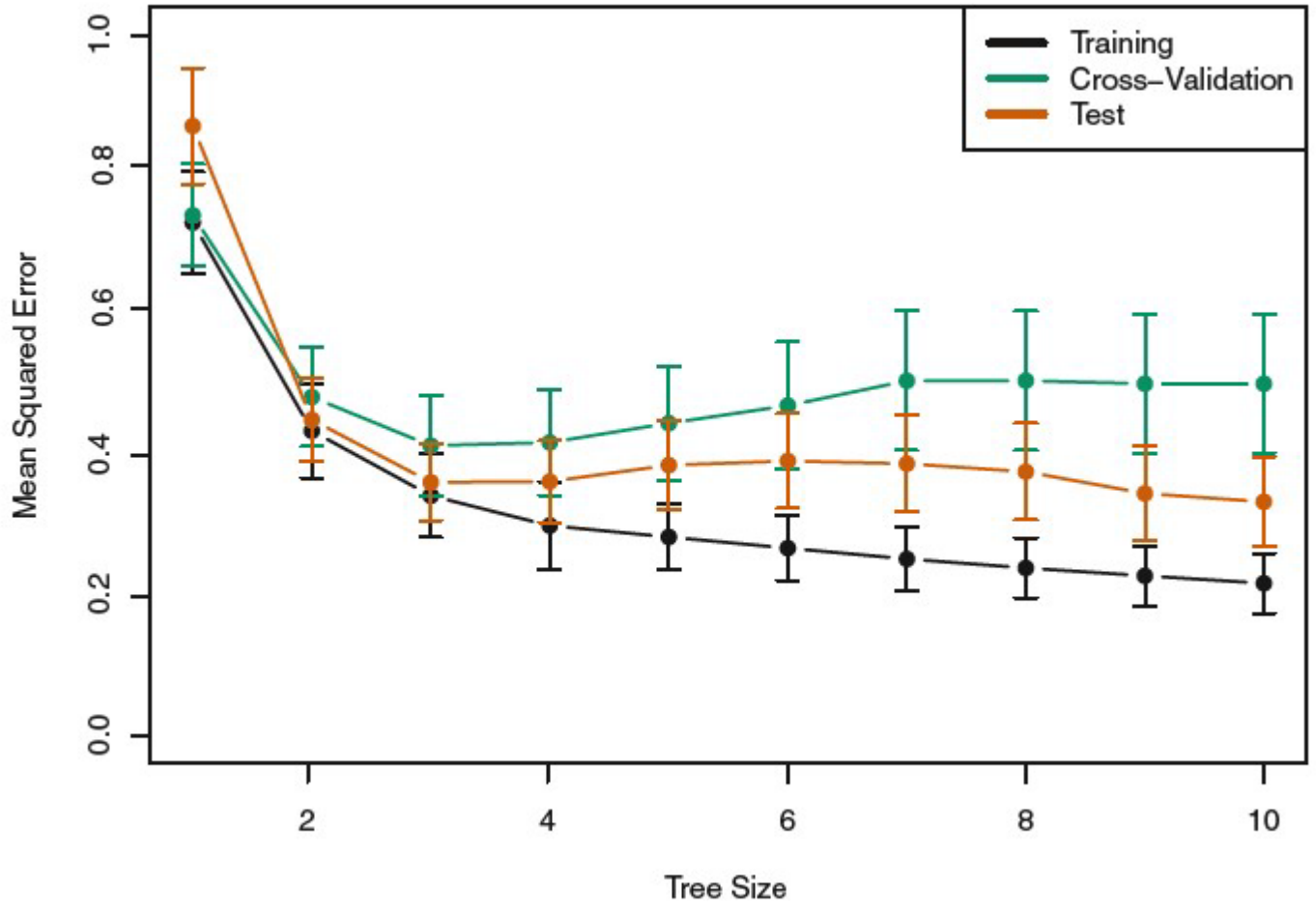
Unpruned Hitters Tree

In this tree note that the same predictor, "Years" is used 4 different times.



Regression Tree: CV analysis

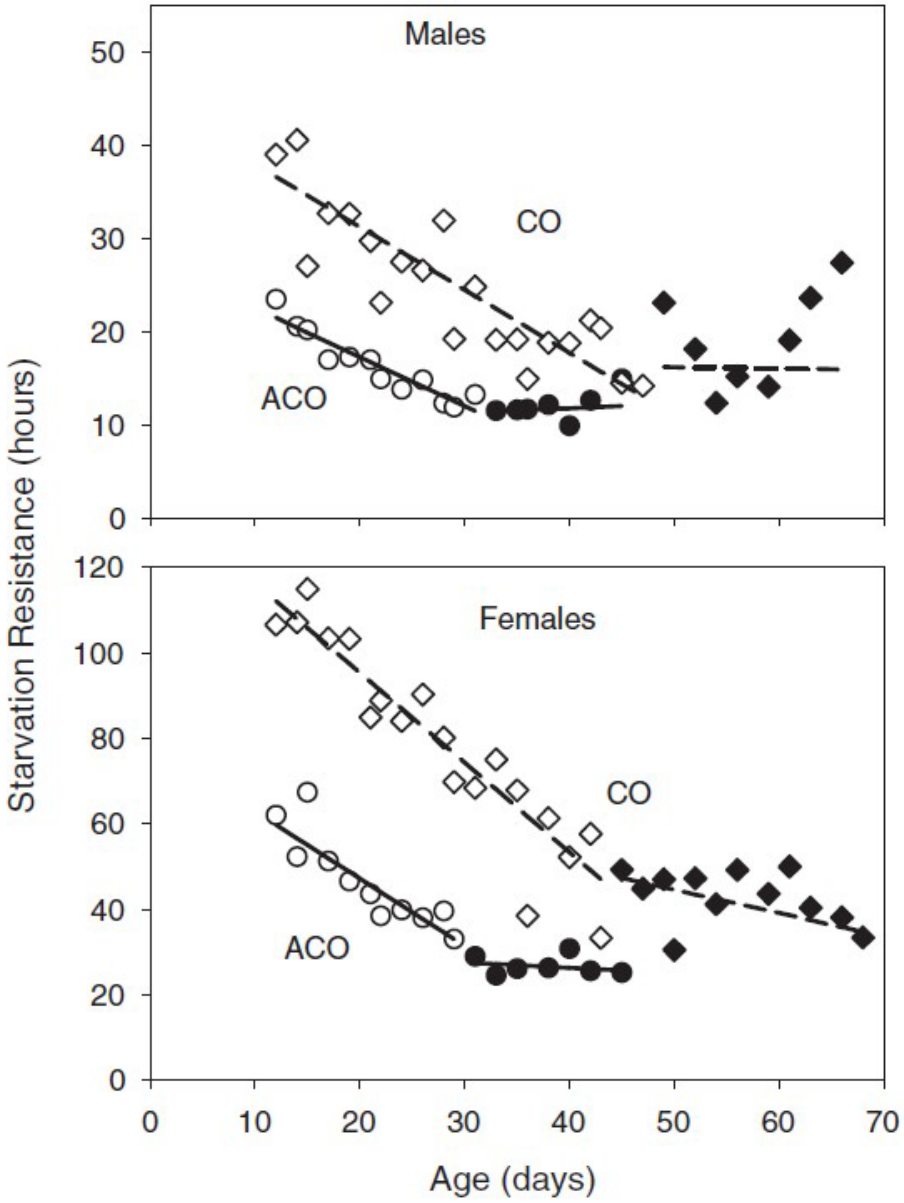
The best tree has three terminal nodes.
This happens to be the first tree shown.



Example: Age specific change in physiology

Shahrestani, P., J. Quach, L.D. Mueller and M.R. Rose. 2012. Paradoxical physiological transitions from aging to late life in Drosophila. Rejuvenation Research.
DOI: 10.1089/rej.2011.1201

Late-life can be defined to begin at the age which mortality plateaus. Is there any value to breaking up physiology into an aging and late-life phase? What if it allows for a better statistical description of the changes in physiology.



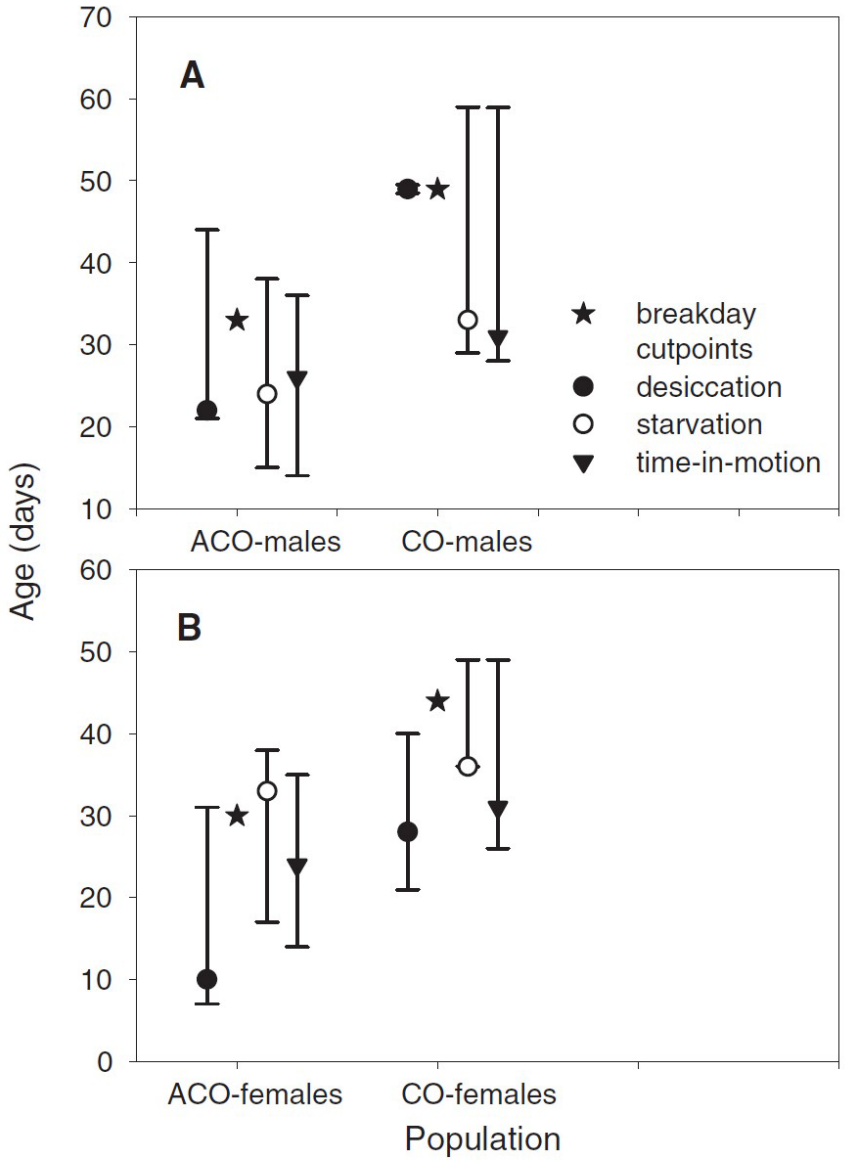
Example: Age specific change in physiology

Physiological measurements, y_{t_i} at ages, $t_0, \ldots, t_n$
Create two regions, R_1={all t_i<t^*} and R_2={all $t_i \geq t^*$}
Find the t^* that minimizes

$$\sum_{i:\, t_i \in R_1} \left(y_{t_i} - \hat{y}_{R_1}\right)^2 + \sum_{i:\, t_i \in R_2} \left(y_{t_i} - \hat{y}_{R_2}\right)^2$$

where $\hat{y}_{R_j}$ is the fitted linear equation $\hat{\beta}_0 + \hat{\beta}_1 t_i$

Confidence intervals were generated from 1000 bootstrap samples. The samples preserved the total sample size and the sample size at each age.



Example: Age specific change in physiology

Data:

```
 co.sel co.pops age sex starv co.plat
24    ACO       1   3    F  74.5     pre
25    ACO       1   3    F  42.5     pre
26    ACO       1   3    F  70.5     pre
27    ACO       1   3    F  74.5     pre
28    ACO       1   3    F  66.5     pre
```

```
library(boot)
breakday<- function(physio.data,index){
  boot.data<- physio.data[index,]
  ages.all<- sort(unique(boot.data$age))
  ages<- ages.all[3:(length(ages.all)-2)]
  rss.age<- NULL
  for (i in ages) {
    temp.a<- boot.data[boot.data$age<i,]
    temp.b<- boot.data[boot.data$age>=i,]
    temp.a.lm<- lm(time~age,data=temp.a)
    temp.b.lm<- lm(time~age,data=temp.b)
    rss.age<- rbind(rss.age,c(sum(temp.a.lm$residuals^2)+ sum(temp.b.lm$residuals^2),i))}
  rss.min<- rss.age[which(rss.age[,1]==min(rss.age[,1])),2] #this gives the age of the minimum
  return(rss.min)
  }
co.m.boot<- boot(physio.data,breakday,R=1000,strata=physio.data[,3])
```

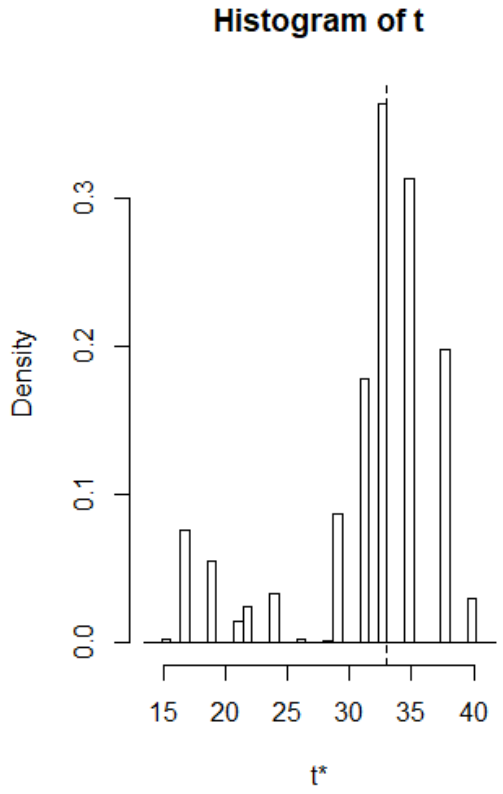
Example: Age specific change in physiology

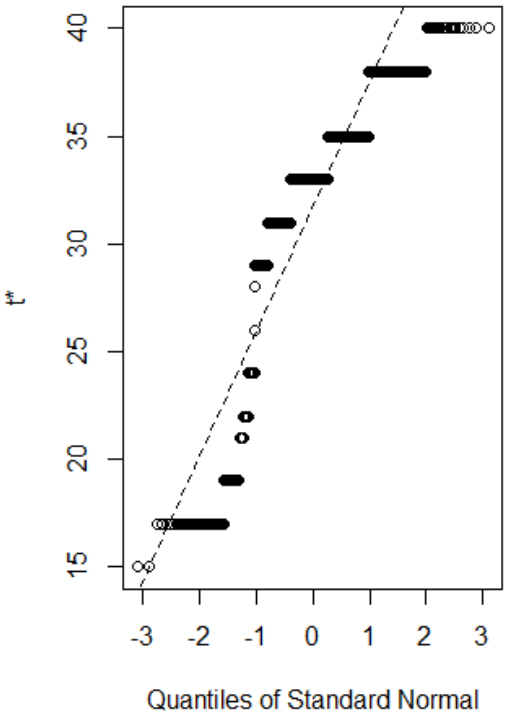
```
> aco.f.boot
STRATIFIED BOOTSTRAP
Call:
boot(data = physio.data, statistic = breakday, R = 1000, strata = physio.data[,
    3])
Bootstrap Statistics :
    original  bias    std. error
t1*       14   0.848    2.274658

> boot.ci(aco.f.boot,type="perc")
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = aco.f.boot, type = "perc")

Intervals :
Level     Percentile
95%   (12, 19 )
Calculations and Intervals on Original Scale
plot(aco.f.boot)
```





Classification Trees

- Here the members of a region will be assigned the class membership of the most commonly occurring class among the training data.
- Recursive binary splitting is used to grow the tree but the RSS can not be used as a criteria.
- One possible criteria is the classification error rate, or the fraction of training observations in a region that do not belong to the most common class.
- Let $\hat{p}_{mk}$ be the proportion of training observations in the *m*th region that are from the *k*th class, then the misclassification error is $1 - \max_k(\hat{p}_{mk})$.
- This measure is not sufficiently sensitive for a growing tree.

Classification Trees

- Let $\hat{p}_{mk}$ be the proportion of training observations in the *m*th region that are from the *k*th class, then the misclassification error is $\mathrm{E} = 1 - \max_k(\hat{p}_{mk})$.
- This measure is not sufficiently sensitive for a growing tree.
- Gini index: $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
- *G* is small when $\hat{p}_{mk}$ is close to 0 or 1, e.g. most observations are of one type, also called node purity.
- Another option is the cross-entropy: $D = -\sum_{k=1}^{K} \hat{p}_{mk}\log(\hat{p}_{mk})$
- The cross-entropy is also close to 0 when $\hat{p}_{mk}$ is near 0 or 1.
- In the growing tree the splits which produce the smallest Gini index or cross entropy will be chosen.
- When pruning the tree use classification error since we typically seek a tree with the best prediction error.

Classification Trees

Patients with chest pain either have heart disease (yes) or don't (no).

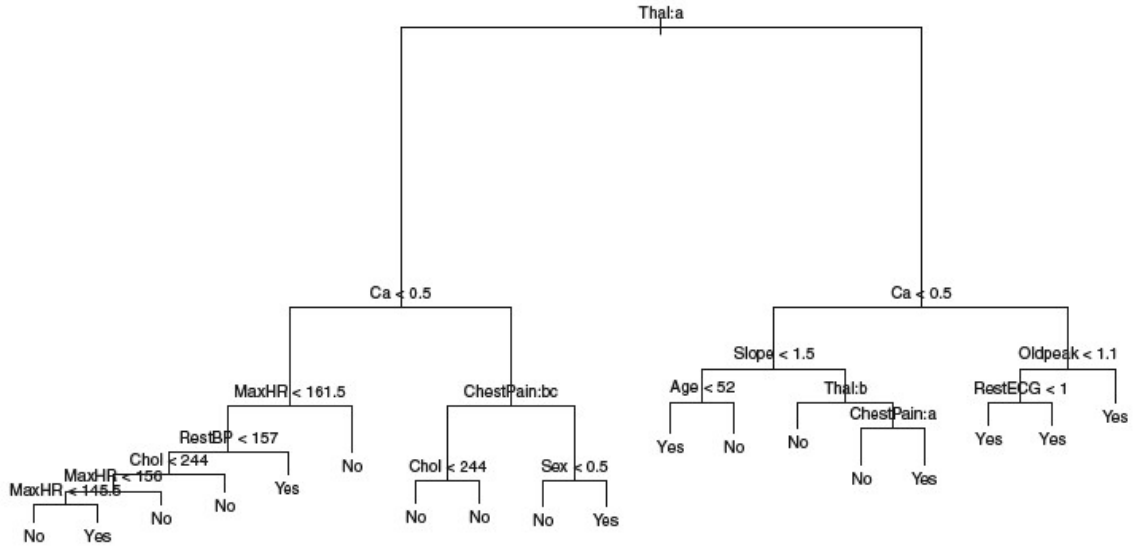
Data base of 303 patients and 13 predictors.

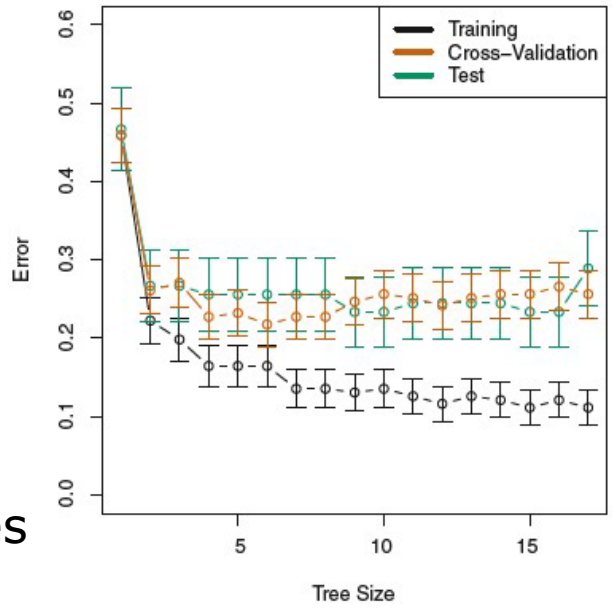
Cross-validation results in a tree with six terminal nodes.

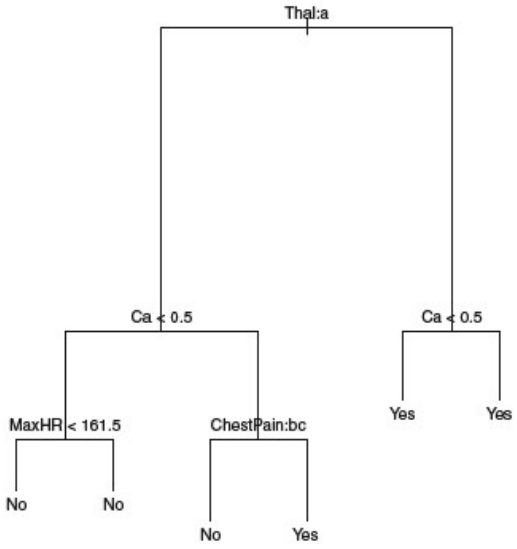
Some of the predictors are qualitative, sex, chest pain. Where these are used at nodes they split the data into two groups.

Some splits results in two nodes with the same prediction. This is due to increase in node purity. So even though the qualitative outcome is the same the node purity is different.

In the right RestECG<1 node all 9 observations are yes. In the left node 7/11 are yes. This split doesn't improve classification error but it improves the Gini index.





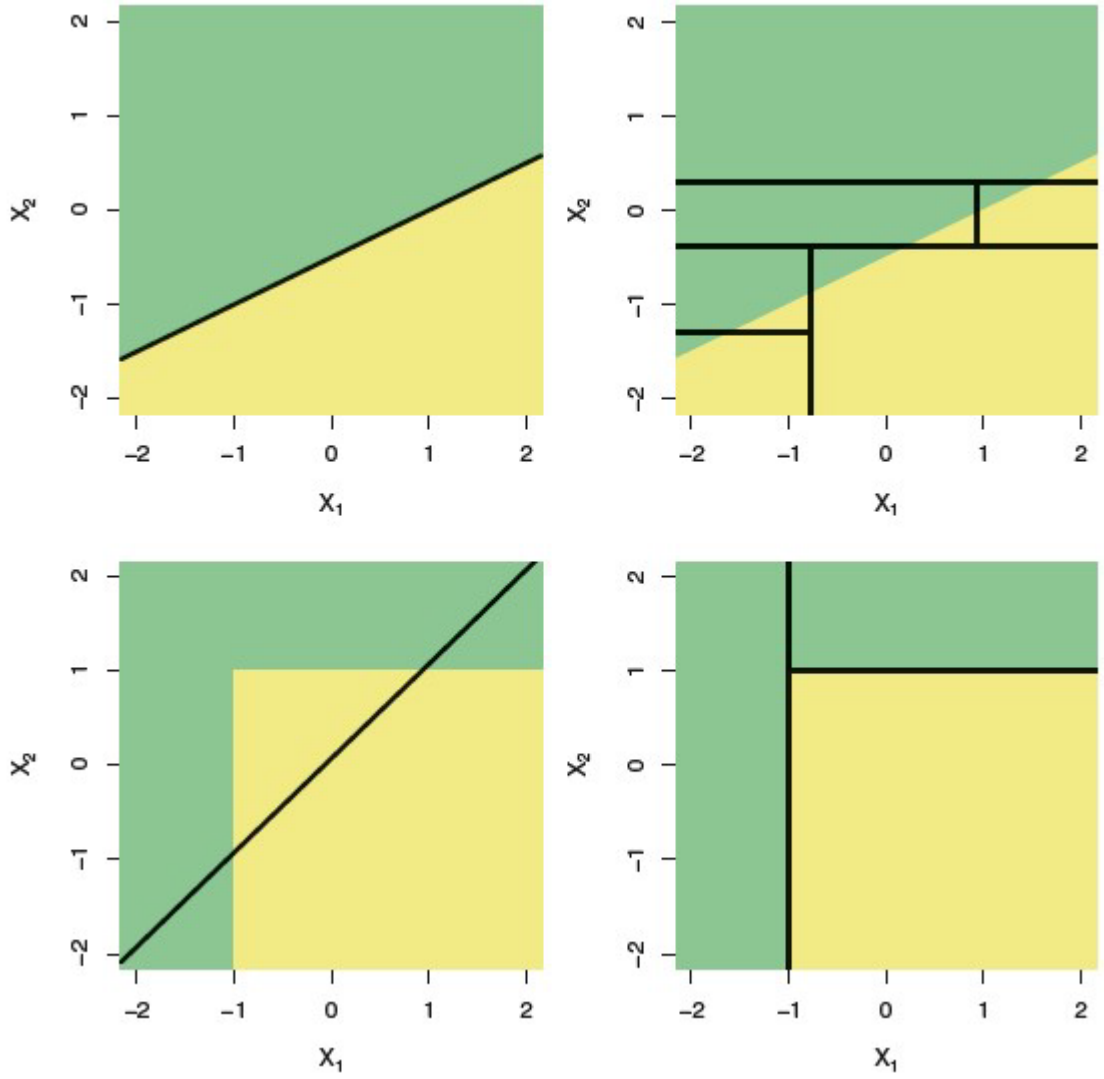


Trees vs. Linear Model

If outcome has a linear relationship between the predictors as in the top panels the linear model will have a lower classification error.

The bottom panel shows a non-linear relationship and the tree does a better job of prediction.

Trees are easy to explain and can be presented graphically.



Bagging

- The process of creating a single tree with recursive binary splitting and cost complexity pruning results in trees with high variance.
- If we could average in some sense many trees we would expect the average predictions based on many trees to have lower variance than the predictions from a single tree.
- One way to accomplish this is the generate many trees from the same data via bootstrap samples.
- Suppose the *b*th bootstrap sample results in model predictions based on the observed vector of predictors, $\boldsymbol{x}$, $\hat{f}^{*b}(\boldsymbol{x})$.
- Then the bagged predictions are,

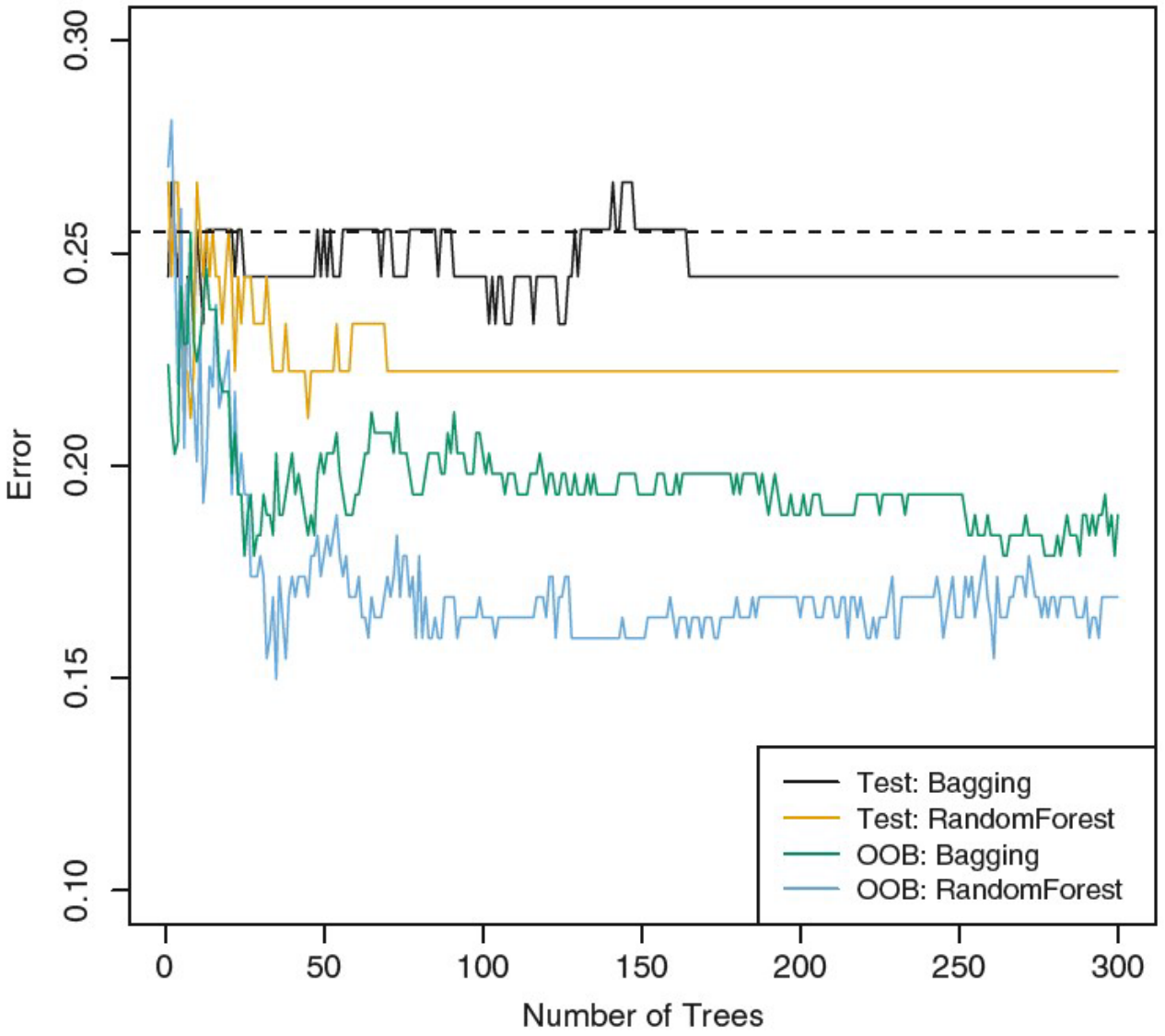
$$\hat{f}_{bag}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(\boldsymbol{x})$$

- For classification problems the bagged estimate is the most common class among the B predictions.

Bagging

- With bagging the trees are grown deeply but are NOT pruned. Thus, they have low bias but high variance. The averaging will hopefully take care of the variance.
- The number of bootstrap trees is generally large in the 100's or thousands.
- The number needed can be determined by examining the behavior of the test error vs the number of samples.

Heart data at:
https://archive.ics.uci.edu/ml/datasets.html



Bagging: Out-Of-Bag (OOB) Error Estimates

- Recall from the prior calculation the chance that a database sample is not included in a bootstrap sample is $(1\text{-}1/n)^n$, where n is the bootstrap sample size.
- This approaches about 1/3. Thus, when constructing the bagged trees there is always about 1/3 of the observations that are not used in the tree construction. These observation are called out-of-bag (OOB) observations.
- These observation can then be used to estimate prediction error.
- The OOB estimates of MSE are valid estimates of the test error.
- See example in the previous slide.

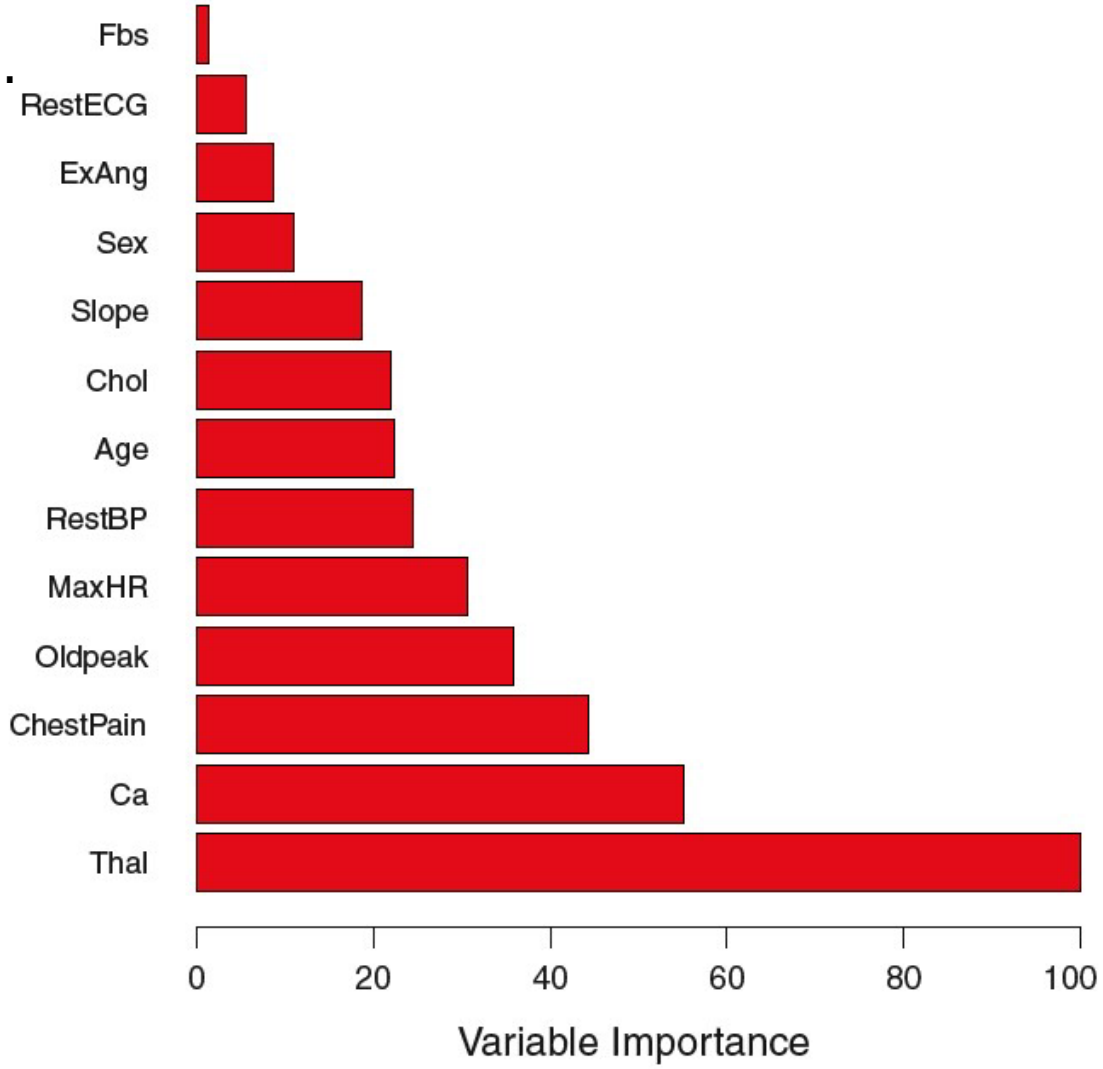
Bagging: variable importance measures

- Although bagging improves prediction accuracy we no longer have a simple tree that can display the results.
- However, we can get an indication of how important each predictor is.
- In the case of a regression tree we can determine how much each predictor has reduced the RSS at each split it was used, over all bootstrapped trees.
- For a classification tree we can gather the same information for the decrease in the Gini index.

Bagging: variable importance measures

Heart data and Gini index ranking of predictors.

The most important predictor, Thal, is given an importance of 100 and all others are ranked relative to Thal.



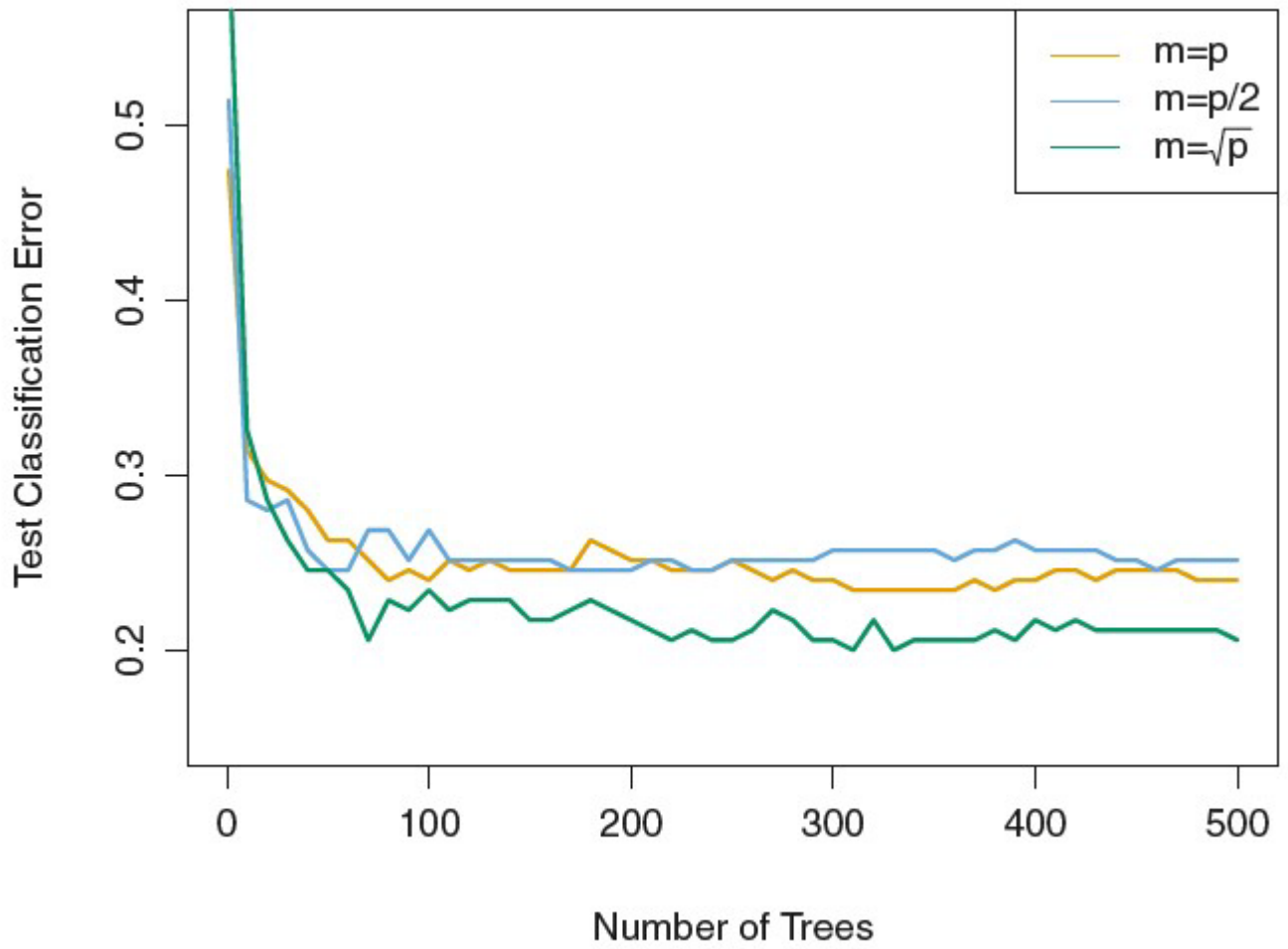
Random Forests

- If there are some very strong predictors it is possible that many of the bootstrap trees will follow at least the same initial structure – making them highly correlated.
- Averaging the predictions of many highly correlated trees will not reduce variance as much as desired.
- To break up these correlations each time a split is considered a random sample of *m* predictors is chosen and the split must be based on these *m* predictors.
- This process should decorrelate the trees and the average predictions should be less variable.
- As a rule of thumb we can set $m = \sqrt{p}$.

Random Forests

Prediction of 14 types of cancer based on expression arrays of 500 genes. These 500 were chosen from a 4,718 based on the magnitude of their variance.

Test error gets appreciably better with $m = \sqrt{p}$ compared to the other two choices.



Boosting

- Boosting is a general technique that can be applied to other methods besides regression and classification trees.
- The boosting process will grow a tree sequentially.
- It also involves a shrinking index which slows the growth of the tree.
- Performance is enhanced by letting the tree “slowly learn”.
- At each of *B* steps in the growing process only a few splits are added to the tree. In fact, often there may be only a single split at each round of growing.

Boosting

Algorithm: boosting regression trees

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:

(a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).

(b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

(c) Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$

3. Output the boosted model,

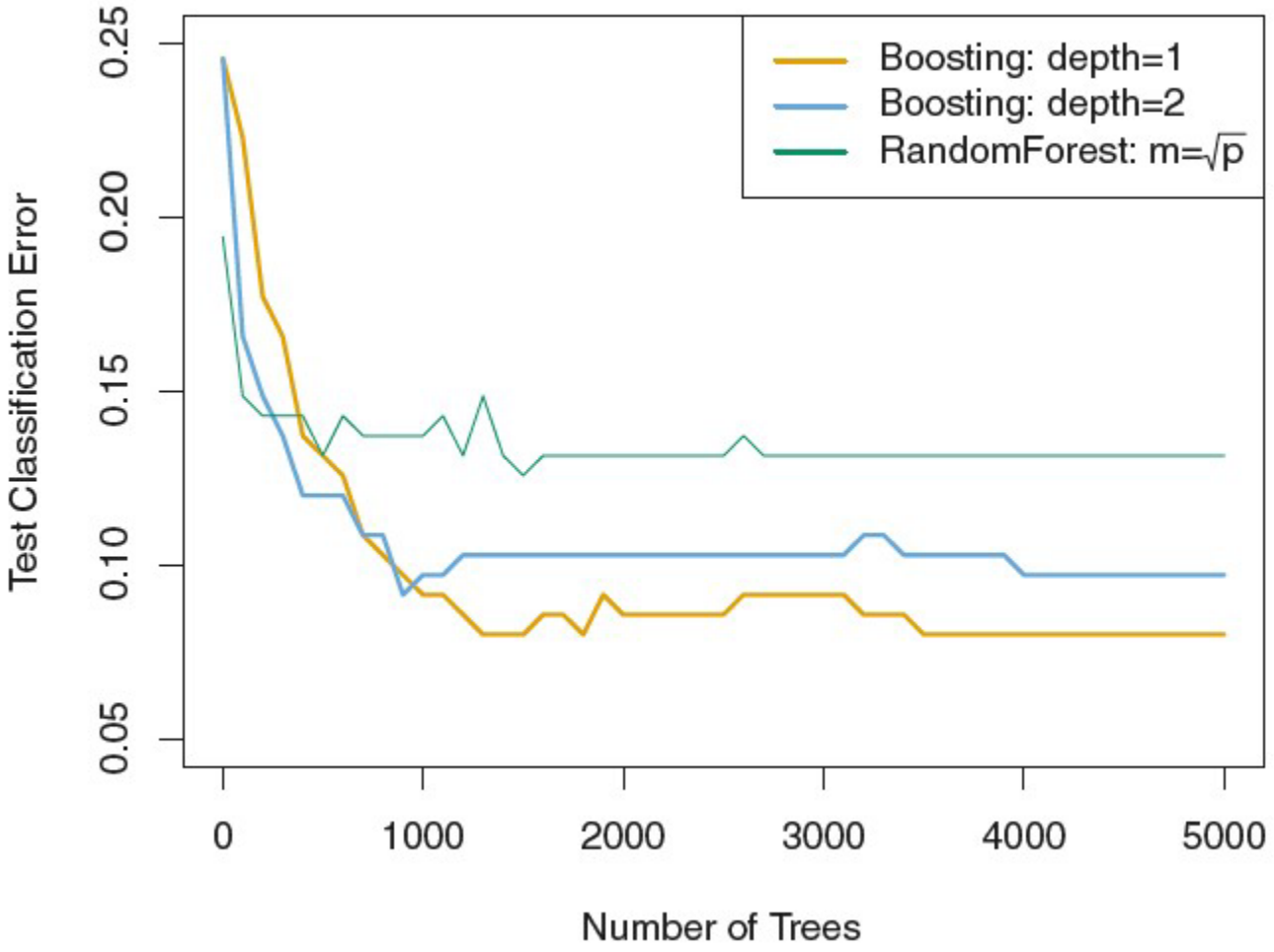
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

Boosting

- Boosting has three tuning parameters.
  1. The number of trees *B*. Boosting can be overfit if B is too large, although this generally happens slowly. Use cross-validation to select *B*.
  2. The shrinkage parameter λ. Typical values are 0.01 or 0.001. Small values of λ will require very large values of *B*.
  3. The number of *d* split in each tree. Often $d=1$ will work well. With $d=1$ the final model is completely additive. The parameter *d* is also referred to as the interactive depth.

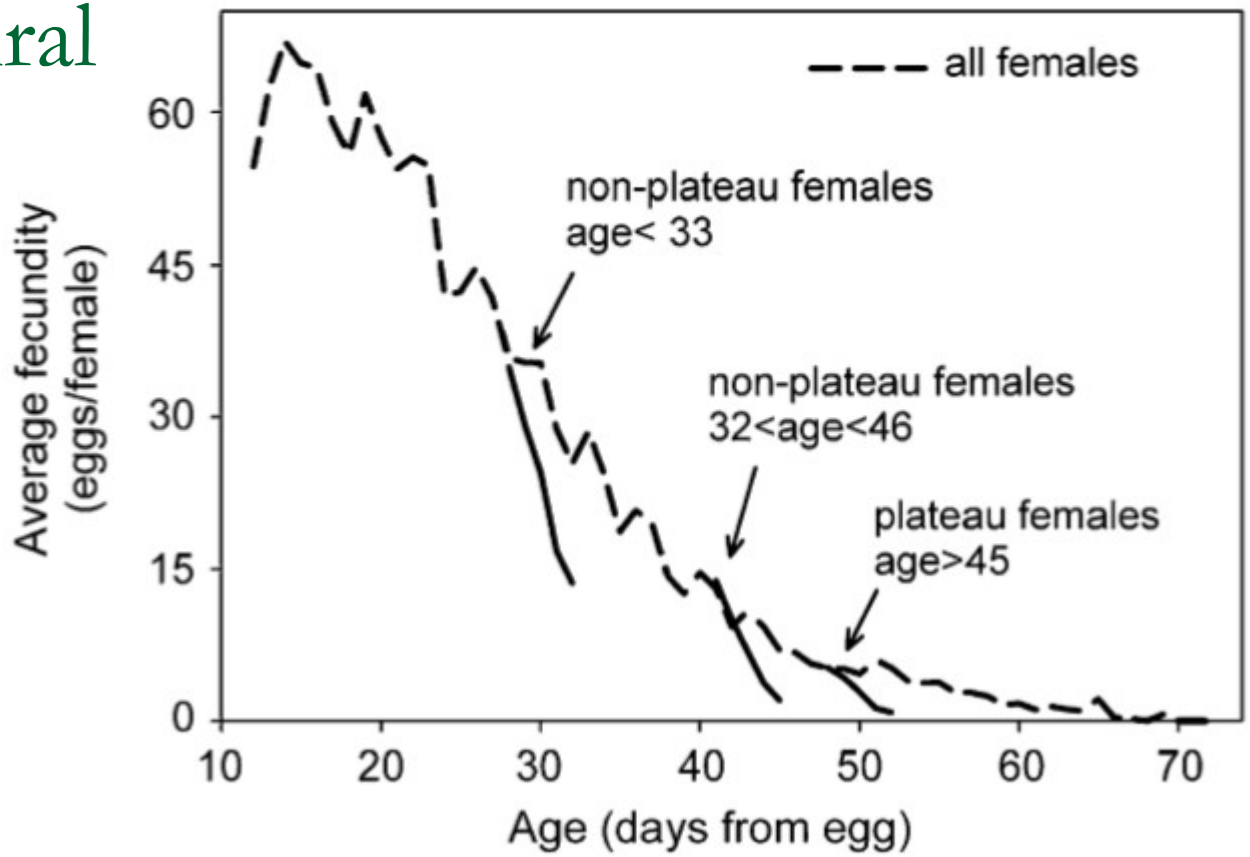
Boosting

This is the cancer data set with the previous random forest results along with boosting at two different depths.



Boosting: Example, death spiral

- Females close to death experience a decline in fecundity at a rate faster than identically aged females not about to die.
- To study this we use scaled fecundity records



Four-day window analysis of data from a single female

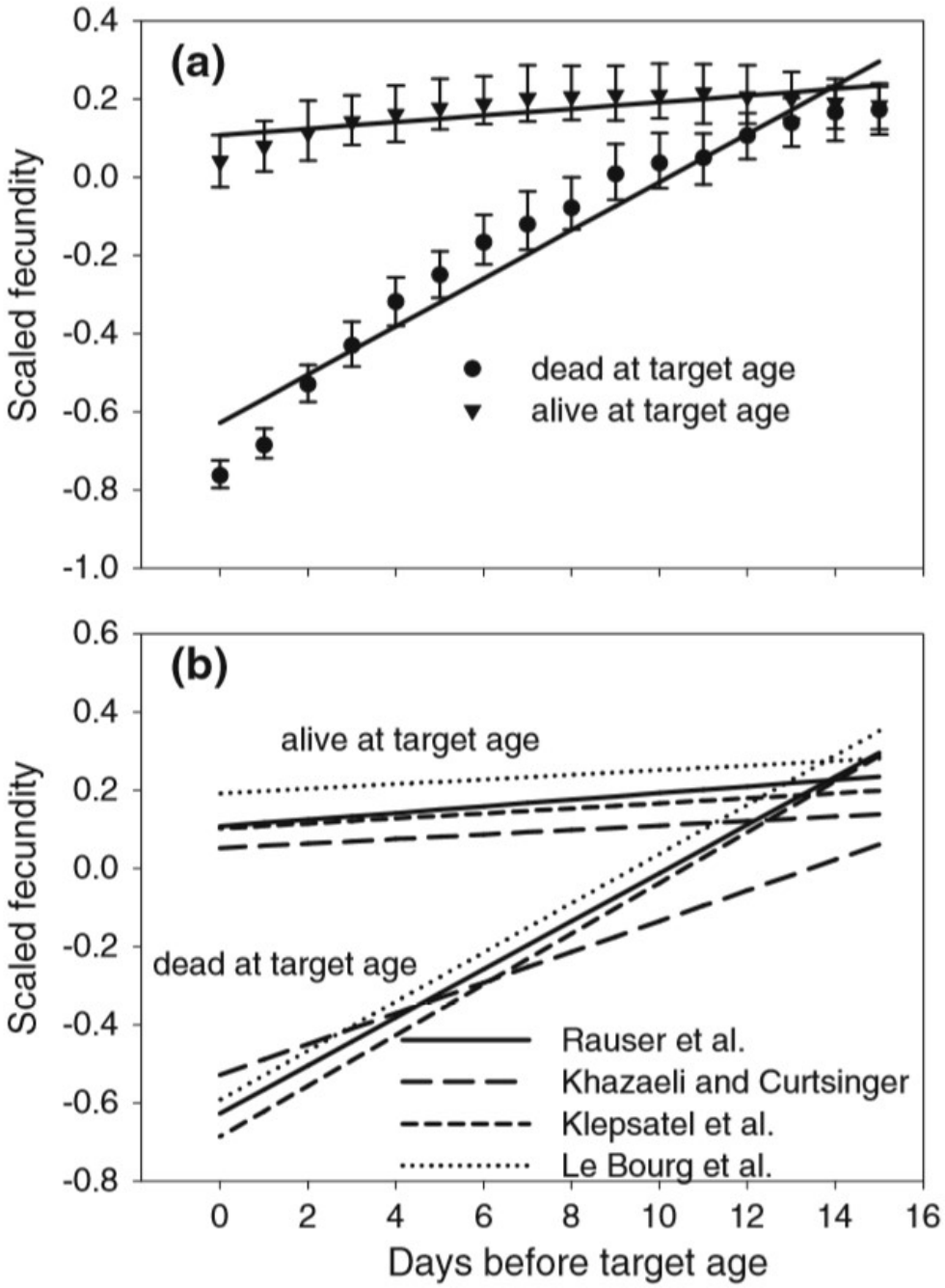
				Target Ages			
Adult age	21	22	23	24	25	26	27
Scaled fecundity	0.9	1.3	0.4	0.1	-1.2	-1.7	NA
Female status	live	live	live	live	live	dead	

A
B
C

Window A. live female status, (0.9, 1.3, 0.4, 0.1)
Window B. live female status, (1.3, 0.4, 0.1, -1.2)
Window C. dead female status, (0.4, 0.1, -1.2, -1.7)

Boosting: Example, death spiral

- The scaled records show a large difference between the flies that die on the target day up to about two weeks before death.
- Can we use the scaled fecundity with individual females to predict which female is about to die?

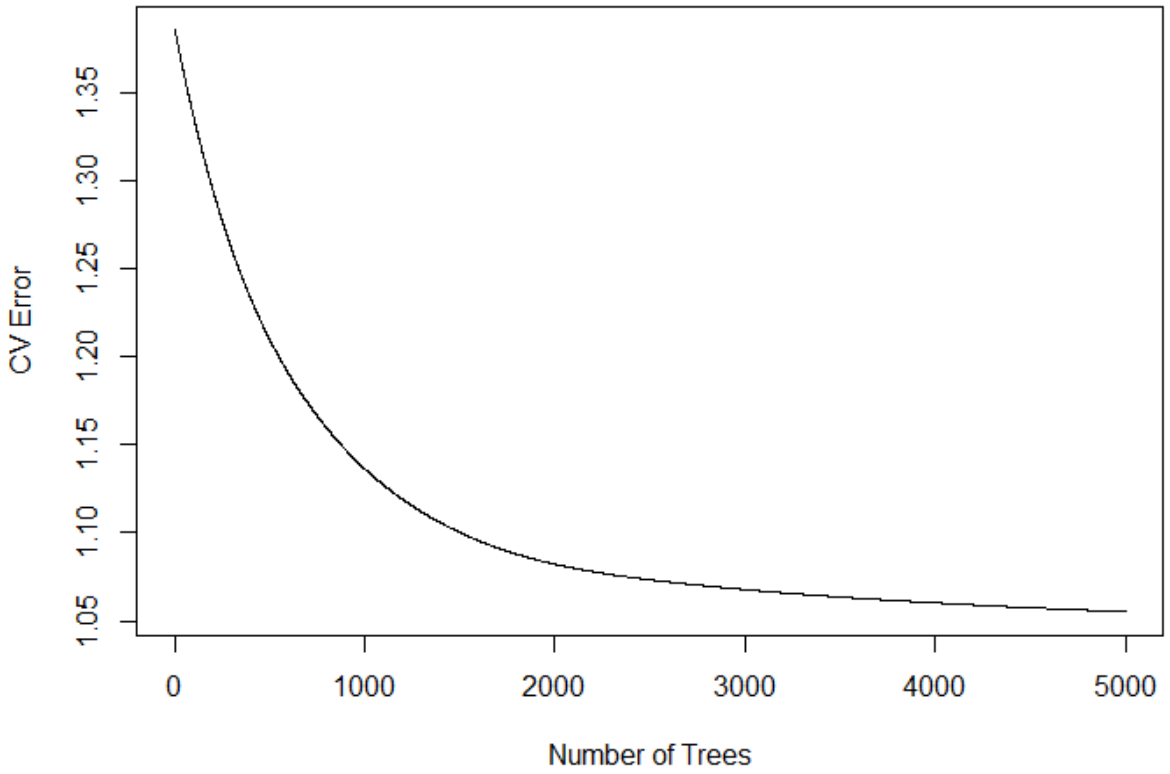


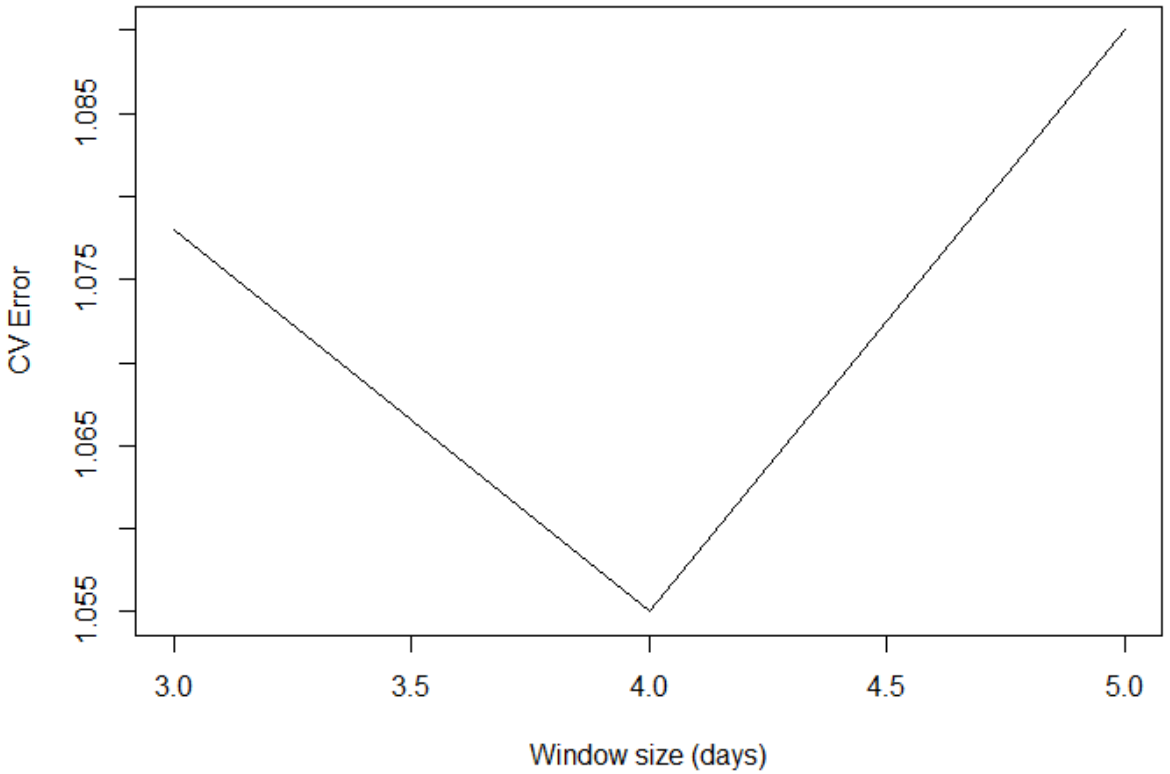
Mueller, L.D., P. Shahrestani, C.L. Rauser, and M.R. Rose. 2016. The death spiral: predicting death in *Drosophila* cohorts. *Biogerontology* DOI 10.1007/s10522-016-9639-7

Boosting: Example, death spiral

```
# We make a database with an equal number of live and dead females, N= 5310.
Of these 1062 are placed in a test database and 4,248 in the training database.
train.5.new<- sample(5310, 4248)
# Below is a sample of the database showing days 1-4 before the target day
 alive       fec5         fec4       fec3       fec2
626      0 -0.3416693 -0.09167779 -0.7241539 -0.4605525
660      0 -2.2190503 -3.20058715 -2.8641337 -2.7655403
727      0  0.1746105  0.66047771  1.0436555  1.1990387
728      0 -0.2008657  0.86105250  0.3923573  0.3692431
1008     0  0.5500867 -2.94986865 -2.8641337 -2.7194405
#Next, we determine the number of trees to use with this 4-day database
library(gbm)#generalized boosted regression modeling
co.5.4.boost<-
gbm(alive~.,data=co.5.4.bdata[train.5.new,],distribution="bernoulli",
interaction.depth=1,n.trees=5000,shrinkage=0.001, cv.folds=10)# Use Bernoulli
                                                              # for 0/1 response
min(co.5.4.boost$cv.error)#= 1.055217
plot(1:5000,co.5.4.boost$cv.error,type= "l",xlab="Number of Trees",ylab="CV
Error")
```

Boosting: Example, death spiral

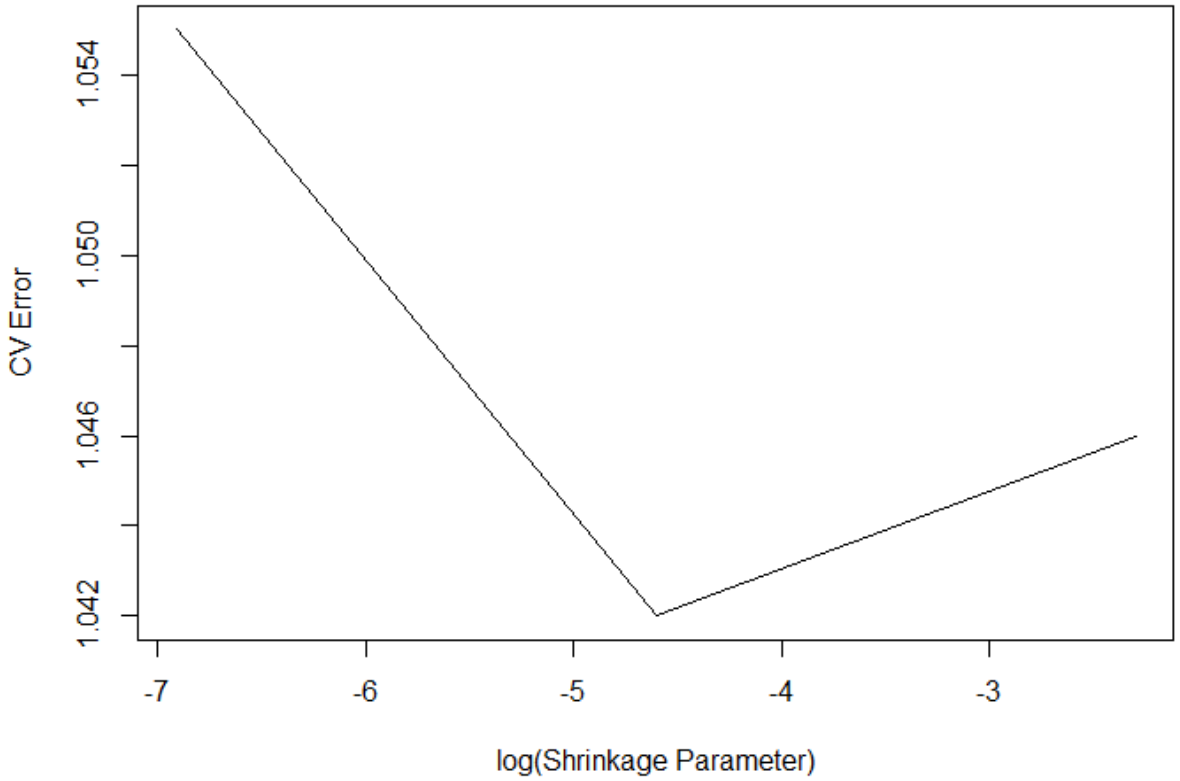




The left graph shows that the CV error is relatively flat at 5,000 trees. The right figure shows that the smallest CV error is with four days of information prior to the target age. The code on the previous page was run with a data base of 3 days of fecundity data and 5 days.

Boosting: Example, death spiral

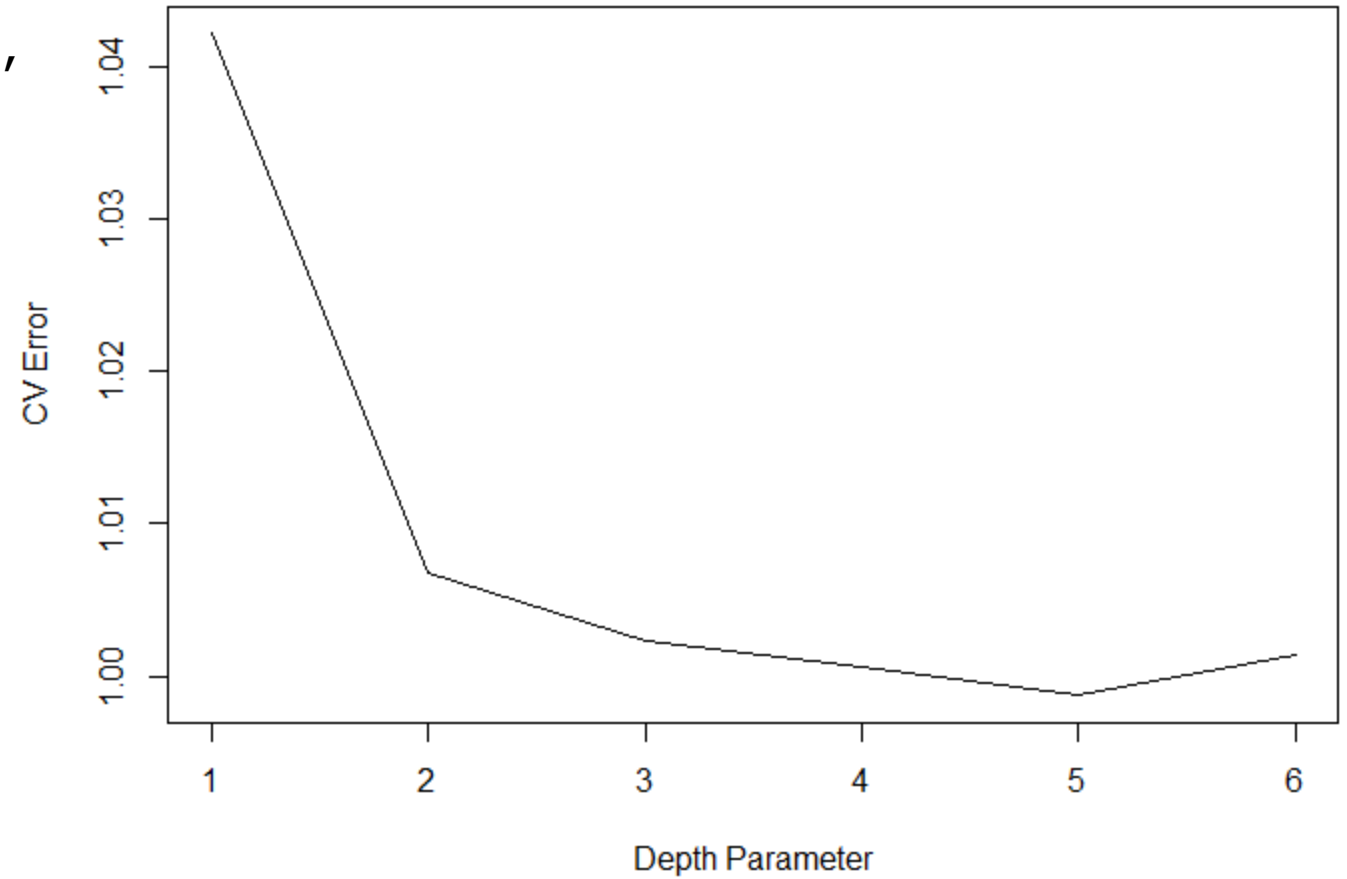
Next, we determine the best value of shrinkage, λ, by examining the CV error with 5,000 trees. So, the best value is λ=0.01



Boosting: Example, death spiral

Next, we tune the depth of each of the 5,000 trees. We see the best depth is five splits, much higher than the books' recommended 1 or 2.

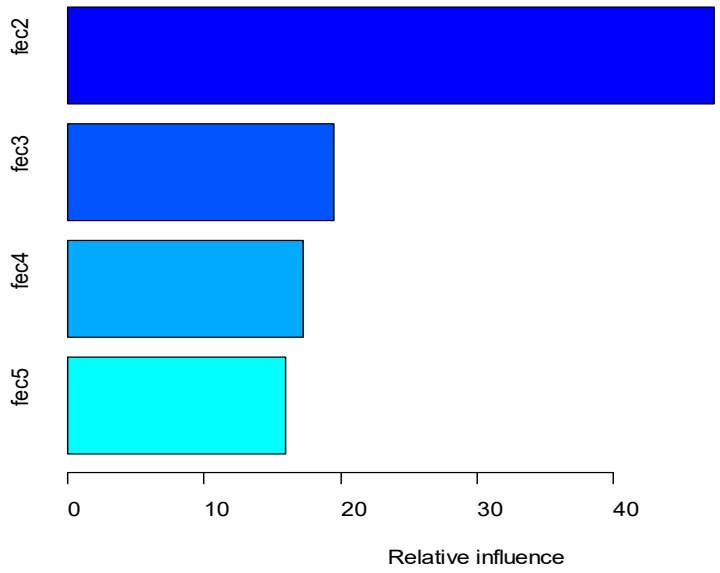
So, the final analysis will use 5,000 trees, λ=0.01, and an interactive depth parameter of 5.



Boosting: Example, death spiral

```
co.5.4.boost.01.d5<- gbm(alive~.,data=co.5.4.bdata[train.5.new,],
distribution="bernoulli",interaction.depth=5,n.trees=5000,shrinkage=0.01,
cv.folds=10)
```

```
>summary(co.5.4.boost.01.d5)
      var  rel.inf
fec2 fec2 47.35058
fec3 fec3 19.49087
fec4 fec4 17.21937
fec5 fec5 15.93918
```



Boosting: Example, death spiral

Finally, we use test data not used in the fitting to predict death and calculate error rates.

```
#co.5.4.bdata[-train.5.new,] is the original database with the training
# data removed -> leaving the test data.
co.5.4.boost.01.d5.pred<- predict(co.5.4.boost.01.d5,
newdata= co.5.4.bdata[-train.5.new,],n.trees=5000,type="response")
temp<- co.5.4.bdata[-train.5.new,1]# temp is just the vector of observed
                                   # status value
table(round(co.5.4.boost.01.d5.pred),temp)
      0   1
  0 421 134
  1 114 393
# Error rate (dead) = 114/( 114+ 421) = 0.2130841 [c.i. 0.1791163 0.2502568]
# Error rate (alive) = 134/(134+393)= 0.2542694 [c.i. 0.2176098 0.2937106]
# These confidence intervals are from binom.test
> binom.test(114,114+421)
        Exact binomial test
data:  114 and 114 + 421
number of successes = 114, number of trials = 535, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.1791163 0.2502568
sample estimates:
probability of success
            0.2130841
```