# Integrating the code and text

*A, A, & R*

*10 March, 2017*

## Contents



## General set up for Aim 2b:

We assume we have a training and a test set. From here forward the training set, $(Z_i, W_i), i = 1 \dots n$, will be used for model building and the test set will solely be used for prediction error once we have a final model. Starting with all of the training data, under $L_2$ loss, both *partDSA* and CART would ordinarily seek to minimize

$$\min_{c_L} \sum_i I\{W_i \in Q_L(j,s)\}(Z_i - c_L)^2 + \min_{c_R} \sum_i I\{W_i \in Q_R(j,s)\}(Z_i - c_R)^2 \quad (1)$$

over all variables $j$ and split points $s$, where $Q_L(j,s) = \{W|W_j \leq s\}$ and $Q_R(j,s) = \{W|W_j > s\}$. The $(j,s)$ combination minimizing (1) can be determined quickly; in CART, the "best" choice is used to divide the root node into two daughter nodes and the above splitting process is then repeated within each daughter node. *partDSA* proceeds similarly, making use of (1) in the addition and substitution steps.

Here, we assume that our training set has been split into two independent subsets: a learning set, $(Z_{0i}, W_{0i}), i = 1 \dots n_0$; and, an evaluation set, $(Z_{1i}, W_{1i}), i = 1 \dots n_1$. We first apply an aggregate learner (e.g., $partDSA_{RF}$) to the learning set $(Z_{0i}, W_{0i}), i = 1 \dots n_0$, generating the (black box) prediction rule, $\hat{m}(w)$; then, we calculate predicted outcomes based on the covariates in the evaluation set, $\widehat{Z}_{1i} = \hat{m}(W_{1i}), i = 1 \dots n_1$. Importantly: $\hat{m}(w)$ and $(Z_{1i}, W_{1i}), i = 1 \dots n_1$ can be considered independent; in addition, the predictions $\widehat{Z}_{1i}, i = 1 \dots n_1$, only utilize the $W_{1i}$'s and not the $Z_{1i}$'s. Importantly, $\widehat{Z}_{1i}, i = 1 \dots n_1$, will not be used as covariates but rather in creating a target for shrinkage, thereby reducing variance.

Strategies 1 & 2 are discussed in the grant application and represent methods for modifying the loss function (1) used for split determination, by making use of the $\widehat{Z}_{1i}$'s to help guide splitting decisions. Here we provide further details that underpin Strategy 2.

## Strategy 2

In addition to $\widehat{m}(\cdot)$, the ensemble building process provides (i) a measure of prediction error, $\hat{\sigma}_0^2$, derived from the learning set, $(Z_{0i}, W_{0i}), i = 1 \ldots n_0$; and, (ii) $B$ predicted outcomes for each evaluation set $W_{1i}$, generating both a predicted mean, $\widehat{Z}_{1i}$, and measure of variance, $\hat{\gamma}_i$. Strategy 2 leverages this information through penalization; the crux of this proposal involves replacing, for $v \in \{L, R\}$, the two optimization problems in (1) with

$$\min_{c_v} \textstyle\sum_i I\{W_{1i} \in Q_v(j,s)\} \left[(Z_{1i} - c_v)^2 + \lambda \alpha_i (c_v - \widehat{Z}_{1i})^2\right] \tag{2}$$

for $\alpha_i > 0$, each having the weighted-average solution

$$\widehat{c}_{v,j,s}(\lambda) = r_{v,j,s}(\lambda)\bar{Z}_{1v}(j,s) + (1 - r_{v,j,s}(\lambda))\hat{\bar{Z}}_{1v}(j,s), \tag{3}$$

where:

$$r_{v,j,s}(\lambda) = 1/(1 + \lambda \bar{\alpha}_{v,j,s}), \tag{4}$$

$$\bar{Z}_{1v}(j,s) = n_{v,j,s}^{-1} \textstyle\sum_i I(W_{1i} \in Q_v(j,s)) Z_{1i}, \tag{5}$$

$$\hat{\bar{Z}}_{1v}(j,s) = \{\textstyle\sum_i I(W_{1i} \in Q_v(j,s))\alpha_i \hat{Z}_{1i}\}/\{\textstyle\sum_i I(W_{1i} \in Q_v(j,s))\alpha_i\} \tag{6}$$

and

$$\bar{\alpha}_{v,j,s} = n_{v,j,s}^{-1} \sum_i I(W_{1i} \in Q_v(j,s))\alpha_i \tag{7}$$

where $n_{v,j,s} = \sum_i I(W_{1i} \in Q_v(j,s))$.

The choice $\alpha_i^{-1} = \text{var}(\hat{Z}_{1i})$ is probably best from an efficiency perpsective; hence the choice of $\alpha_i = \hat{\gamma}_i^{-1}$ in the grant application text. Note that, given $\bar{\alpha}_{j,v,s}$, more weight is placed on $\hat{\bar{Z}}_{1v}(j,s)$ when $\lambda$ is larger; similarly, given $\lambda$, more weight is placed on $\hat{\bar{Z}}_{1v}(j,s)$ when $\bar{\alpha}_{j,v,s}$ is larger. Both make sense in shrinking the parameter estimate towards this weighted mean function.\

To decide:

1. Is this the most appropriate formulation?
2. How to choose $\lambda$.

### Choice of $\lambda$: Within-Node

Applying the results of Section  and assuming all expectation calculations are conditional on $W_{1i}, i \geq 1$ it can be shown that

$$K_2 = n_{v,j,s}^{-1}$$

and

$$K_1 = \hat{\bar{Z}}_{1v}(j,s).$$

Assuming that $E(Z_{1i}) = \mu_{Z_1}$ and $var(Z_{1i}) = \sigma_{Z_1}^2$ when $I(W_{1i} \in Q_v(j,s)) = 1$ (i.e., constant mean and variance within a node), the "best" within-node choice of $\lambda$ via (19) becomes

$$\lambda_{opt} = \frac{n_{v,j,s}^{-1}\sigma_{Z_1}^2}{\bar{\alpha}_{v,j,s}(\mu_{Z_1} - \hat{\bar{Z}}_{1v}(j,s))^2}. \tag{8}$$

Note that selecting

$$\hat{\bar{Z}}_{1v}(j,s) = \{\sum_i I(W_{1i} \in Q_v(j,s))\hat{Z}_{1i}\}/\{\sum_i I(W_{1i} \in Q_v(j,s))\} \tag{9}$$

in equation (8) instead of $\hat{\bar{Z}}_{1v}(j,s)$ (defined in (6)) gives an alternative shrinkage target. There are other choices as well. Thus, Strategy 2 can be viewed as a procedure for shrinking the node-specific estimates towards some node-specific average predicted value.

**Choice of $\lambda$: Prediction Error with Grid**

Instead of using a within-node selection of $\lambda$, we can implement a global method for picking $\lambda$ by minimizing the prediction error over a grid of possible values for $\lambda$. Suppose that $\hat{\mathcal{M}}(W,\lambda)$ denotes the final prediction rule obtained using the data $(Z,W)$ – meaning, this is obtained from our proposed penalized loss procedure for fixed $\lambda$. Let $\mathcal{N}_1(\lambda),\ldots,\mathcal{N}_{K(\lambda)}(\lambda)$ be the partitions obtained in the final structure built with fixed $\lambda$; then, we know

$$\hat{\mathcal{M}}(W_{1i},\lambda) = \sum_{k=1}^{K(\lambda)} I\{W_{1i} \in \mathcal{N}_k(\lambda)\}\hat{c}_k(\lambda) \tag{10}$$

(piecewise constant predictor within each partition/node). Here,

$$\hat{c}_k(\lambda) = r_k(\lambda)\bar{Z}_{1j} + (1 - r_k(\lambda))\hat{\bar{Z}}_{1k}$$

where $r_k(\lambda) = 1/(1+\lambda\bar{\alpha}_k(\lambda))$, $\bar{Z}_{1k}$ is the node-specific mean of the $Z_{1i}$s,

$$\hat{\bar{Z}}_{1k} = \{\sum_i I(W_{1i} \in \mathcal{N}_k(\lambda)\alpha_i\widehat{Z}_{1i}\}/\{\sum_i I(W_{1i} \in \mathcal{N}_k(\lambda))\alpha_i\}$$

and

$$\bar{\alpha}_k(\lambda) = \{\sum_i I(W_{1i} \in \mathcal{N}_k(\lambda))\alpha_i\}/\{\sum_i I(W_{1i} \in \mathcal{N}_k(\lambda))\}.$$

This is a very complicated function of $\lambda$ and the within-node procedure described in Section  probably cannot be directly adapted to choose a global $\lambda$.

Per Efron & Tibshirani (1993) and Efron (2004),

$$\text{err}(\lambda) := \sum_{i=1}^{n_1}(Z_{1i} - \hat{\mathcal{M}}(W_{1i},\lambda))^2 \tag{11}$$

is a version of the "apparent" prediction error because $\hat{\mathcal{M}}(W_{1i},\lambda)$ is built using the data$(Z,W)$. As this is an optimistic assessment of error, we do not want to use it to choose $\lambda$. Following Efron (2004) a preferred measure of error is

$$\text{Err}(\lambda) := E_{Z_{20},W_{20}}\left[(Z_{20} - \hat{\mathcal{M}}(W_{20},\lambda))^2\right]$$

where $(Z_{20},W_{20})$ is independent of $(Z_{1i},W_{1i}), i = 1\ldots n_1$ and $\hat{\mathcal{M}}(w,\lambda)$ is held fixed in the expectation calculation. Calculations in Efron (2004, Eqn. 2.8) show

$$E[\text{Err}(\lambda)] = E[\text{err}(\lambda) + 2cov(Z_{20},\hat{\mathcal{M}}(W_{20},\lambda))];$$

this implies $\text{err}(\lambda) + 2cov(Z_{20}, \hat{\mathcal{M}}(W_{20}, \lambda))$ is an unbiased estimator of $E[\text{Err}(\lambda)]$ (which is just the expected prediction error); here, the covariance term acts as a bias correction. However, except in simple linear smoothing problems, $cov(Z_{20}, \hat{\mathcal{M}}(W_{20}, \lambda))$ is not easy to calculate or otherwise estimate analytically.

Efron (2004) proposes to use a parametric bootstrap procedure to deal with this problem. Again, consider a fixed $\lambda$. Following Efron (2004), suppose we generate the $b^{th}$ bootstrap sample $Z^*_{1i}(b) \sim N(\hat{\mathcal{M}}(W_{1i}, \lambda), \hat{\sigma}^2_{Z_1 - \hat{\mathcal{M}}}), i = 1, \ldots, n_1$, where

$$\hat{\sigma}^2_{Z_1 - \hat{\mathcal{M}}} = n_1^{-1} \sum_{i=1}^{n_1} (Z_{1i} - \hat{\mathcal{M}}(W_{1i}, \lambda))^2. \tag{12}$$

For generating boostrap samples, we can use $\hat{m}(\cdot)$ in place of $\hat{\mathcal{M}}(W_{1i}, \lambda)$, as bootstrapping does not depend on $\lambda$ and this estimation only needs to be done once.

For each $b = 1, \ldots, B$ we run our code on $\{(Z^*_{1i}(b), W_{1i}, \hat{Z}_{1i}), i = 1, \ldots, n_1\}$, to obtain a new $\hat{\mathcal{M}}^*(w, \lambda)$. We can compute for each $i = 1, \ldots, n_1$

$$C^*_i(\lambda) = \frac{1}{B-1} \sum_{b=1}^{B} \hat{\mathcal{M}}^*(W_{1i}, \lambda)(Z^*_{1i}(b) - \bar{Z}^*_{1i}), \tag{13}$$

where

$$\bar{Z}^*_{1i} = \frac{1}{B} \sum_{b=1}^{B} Z^*_{1i}(b) \tag{14}$$

and then define the boostrap corrected error as

$$\text{err}_{cor}(\lambda) = \text{err}(\lambda) + 2 \sum_{i=1}^{n_1} C^*_i(\lambda) \tag{15}$$

If run over a grid of possible $\lambda$ values, it should be possible to choose the $\lambda$ that minimizes $err_{cor}(\cdot)$ (or a smoothed version of it).

- Why are we not bootstrapping the entire training set?

# Code

Code has been written that implements Strategy 2. For the moment we do not have a test set; thus, the entire dataset is the training set with half for the learning set and half for the evaluation set. To choose $\lambda$ we have started with the estimate of $\lambda_{opt}$ in (8) at the root node. We multiply that estimate by a constant $c$ and do a grid search on [0,c$\lambda$]. The final $\hat{\lambda}$ is the one gives the best optimism-corrected error rate in (15). Once we have $\hat{\lambda}$ we build a CART tree based on (2).

## Functions

The goal of this code is to build an interpretable tree by employing the predictive accuracy of a bagged learner.

`aim2`

The main function is the `aim2` function which has calls:

- **dat** is the data frame to which the model is fit
- **nreps** is set to 1 for the time being
- **ngrid** is the number of lambdas in the grid search
- **mult** is the number multiplied times the intial lambda which gives the maximum lambda in the grid search
- **seed** fixes the random number generator for reproducibility
- **outvar** is the name of the outcome variable in the fitting

**The output from the function 'aim2 is:**

- **lambdas** are the values of lambda from grid search
- **Error.lambdas** are the bootstrapped corrected errors from (15)
- **error.lambdas** is the apparent prediction error from (11)
- **optimism** is $2 * \sum_i^{n_1} = 1 C_i^*(\lambda)$ from the right hand side of sum in (15)
- **fits** are the new $\hat{\mathcal{M}}(W_{1i}, \lambda)$ for the different lambdas
- **predictions** are the predicted values for the evaluations set from $\hat{\mathcal{M}}(W_{1i}, \lambda)$
- **evaluation.dat** is the evaluation data

```
aim2 <- function(dat,nreps=1,n.grid=20,mult=2,seed=12345,outvar="Y")
{
  #Functions that go into penalized fitting method
  aim2.list <- list(eval=aim2.eval, split=aim2.split, init=aim2.init,
                    summary=aim2.summary, text=aim2.text)
  set.seed(seed)
  n <- nrow(dat)

  #Identify outcome variable   --- this is redundant with functions statement and should be changed
  which.outcome <- which(colnames(dat)==outvar)
  colnames(dat)[which.outcome] <- "outvar.aim2"

  #Split training set into learning and evaluation sets - now based on 50/50 split
  # later look at different alternatives
  nlearn <- round(n/2)
  neval <- n-nlearn
  samp <- sample(1:n,n,replace=FALSE)
  wlearn <- sort(samp[1:nlearn])
  weval <- sort(samp[(nlearn+1):n])
  learning.dat <- dat[wlearn,]
  evaluation.dat <- dat[weval,]

  #The lambdas chosen using a grid. Here, get values for variables based on root node
  fit.rf.learning <- randomForest(outvar.aim2 ~ .,data = learning.dat) # Fit RF with learning set
```

```
  predict.rf.evaluation <- predict(fit.rf.learning,newdata=evaluation.dat,
                                   predict.all=TRUE) #  $\widehat{Z}_{1i}$
  mean.evaluation <- mean(evaluation.dat$outvar.aim2) # $\mu_{Z_1}$
  var.evaluation <- var(evaluation.dat$outvar.aim2) # $\sigma^2_{Z_1}$
  zbarhat <- mean(predict.rf.evaluation$aggregate) # $ \bar{\hat{Z_1}}$

    # NOTE: this zbarhat means we are doing the
    # alternative shrinkage target in equation 9 not the original in 6

  var.z1s <-  apply(predict.rf.evaluation$individual,1,var) # $\sigma^2_{\hat{Z_1i}}$
  alphas <- 1/var.z1s # $\alpha_i$
  alphabar <- mean(alphas) # \bar{\alpha}$
  lambda <- var.evaluation/(neval*alphabar*(mean.evaluation-zbarhat)^2) #with-in node
                                                                        #choice of \lambda
  lambdas <- seq(0,mult*lambda,length.out=n.grid)  # list of possible lambdas
  n.lambdas <- length(lambdas) #length of list
  error.lambdas <- rep(0,length(lambdas))
  fits <- vector("list",n.lambdas)
  predictions <- vector("list",n.lambdas)

  # To get the err(\lambda) - uncorrected - currently equation 11
  for(j in 1:n.lambdas)
    {
      current.fit <- rpart(outvar.aim2 ~ .,data = evaluation.dat,
                           parms=list(lambda=lambdas[j],
                           yhat=predict.rf.evaluation$aggregate,
                           alpha=alphas),method=aim2.list)
      predicted.fit <- predict(object=current.fit,newdata=evaluation.dat)
      error.lambdas[j] <- sum((evaluation.dat$outvar.aim2-predicted.fit)^2)
      fits[[j]] <- current.fit
      predictions[[j]] <- predicted.fit
  }

  # To get the optimism for correcting the err(\lambda)
  optimism <- corrected.lambda(dat=evaluation.dat,lambdas=lambdas,
                               list.object=aim2.list,model=fit.rf.learning,
                               predicted.values=predict.rf.evaluation$aggregate,
                               alphas=alphas,n.boot=10)

  # err(\lambda) corrected
  Error.lambdas <- error.lambdas+optimism

  #return list of interesting variables.
  list(lambdas=lambdas,Error.lambdas=Error.lambdas,error.lambdas=error.lambdas,
       optimism=optimism,fits=fits,predictions=predictions,
       predicted.fit=predicted.fit,evaluation.dat=evaluation.dat)
}
```

At the end of this function, `optimisim` is as written in equation (13) and `Error.lambdas` is in equation (15).

### Corrected lambda function

This function is called by `aim2` to get the optimism correction for the prediction error. This implements the parametric boostrap and evaluates equations (12) - (14) and returns the righthand side of (15).

**This function assumes that the last column of the data is the outcome – this needs to be fixed Two places: dat[,p] and new.dat[,p] <- boot.dat[,b]**

```
corrected.lambda <- function(dat,lambdas,list.object,model,predicted.values,alphas,n.boot=10)
  {
    n1 <- nrow(dat)
    p <- ncol(dat)
    n.lambdas <- length(lambdas)
    cilambda <- matrix(0,n1,n.lambdas)
    boot.dat <- boot.residual <- matrix(NA,n1,n.boot)
    sigmahat <- sqrt(sum((dat[,p]-predicted.values)^2)/n1) #SD for \hat{\sigma^2_{Z_1-\bigM}}

    for(b in 1:n.boot) boot.dat[,b] <- rnorm(n1,mean=predicted.values,sd=sigmahat)#bootstrap samples
    boot.mean <- matrix(apply(boot.dat,1,mean))  # \bar{Z^*_{1i}}
    for(i in 1:nrow(boot.dat)) boot.residual[i,] <- boot.dat[i,]-boot.mean[i]
    for(b in 1:n.boot)
      {
        new.dat <- dat
        new.dat[,p] <- boot.dat[,b]
        for(j in 1:n.lambdas)
          {
            final.fit <- rpart(outvar.aim2 ~ .,data = new.dat,
                               parms=list(lambda=lambdas[j],
                               yhat=predicted.values,alpha=alphas),
                               method=list.object)
            bigMhat <- predict(object=final.fit,newdata=new.dat)
            cilambda[,j] <- cilambda[,j]+bigMhat*boot.residual[,b]
          }
      }
    cilambda <- cilambda/(n.boot-1)
    return(2*apply(cilambda,2,sum))
  }
```

### Hand build rpart tree

To begin we call needed libraries and code the rpart functions for init, eval, and split which are specific to our algorithm. To build an rpart tree by hand, a list of functions needs to be fed to the rpart call. That list is referred to as aim2.list, and used with the argument method=aim2.list. Important functions are an initialization function (aim2.init), an evaluation function (aim2.eval), and a splitting function (aim2.split). Note that in aim2.init the $y$ variable contains three columns: the evaluation set outcome variables $Z_{1i}$, the $\alpha$'s, and the predicted values $\widehat{Z_{1i}}$. In aim2.eval the value of (2) is computed for the chosen split. In aim2.split the optimal split is found. This is done currently by looping through every value of every variable. Future effort will be undertaken to see if the loop can be removed.

```
#y contains response Z_i, Zhat_i from RF, and alpha where alpha=1/var(zhat)

aim2.init <- function(y, offset, parms, wt)
{
  if (!is.null(offset)) y[,1] <- y[,1]-offset
```

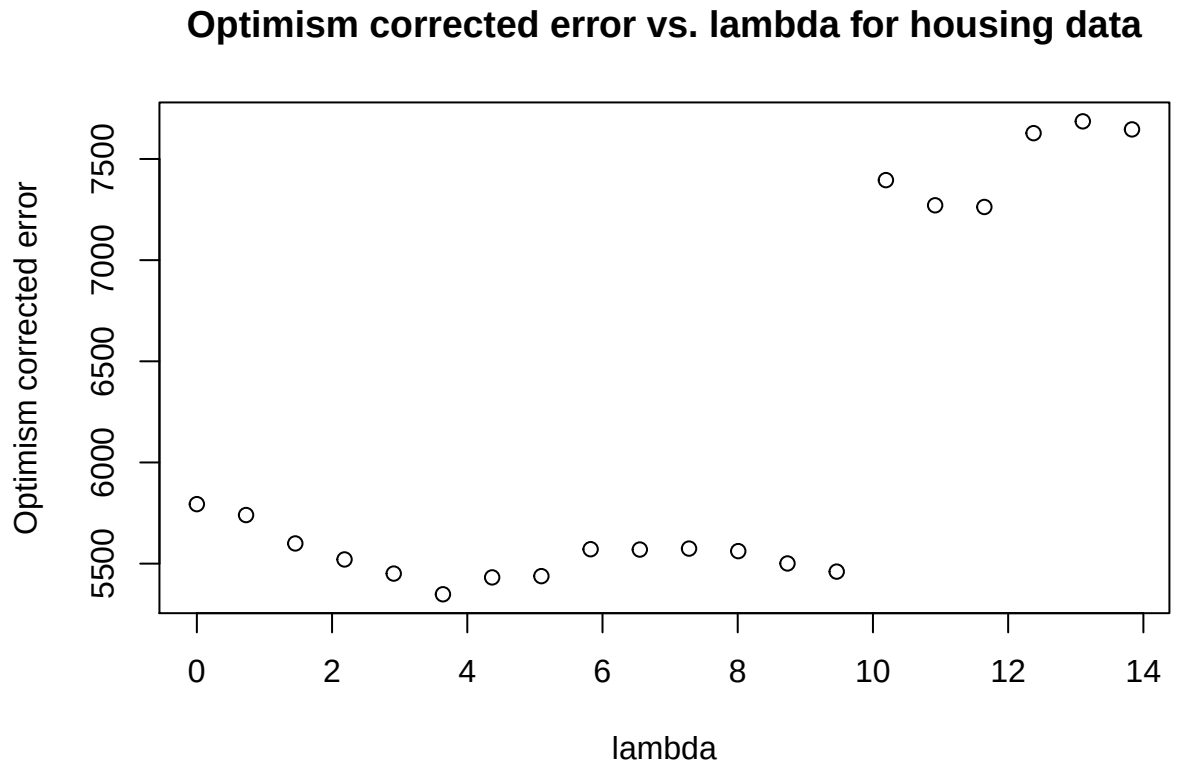
```
  list(y=cbind(y,parms$yhat,parms$alpha),parms=parms, numy=3, numresp=1, summary=aim2.summary)
}

aim2.eval <- function(y, wt, parms)
{
  n <- length(y)/3
  lambda <- parms$lambda
  yhat <- y[,2]
  alphas <- y[,3]
  alphabar <- sum(alphas)/n
  y1 <- y[,1]
  r <- 1/(1+lambda*alphabar)
  zbar <- mean(y1)
  zbarhat <- sum(yhat*alphas)/sum(alphas)
  chat <- r*zbar+(1-r)*zbarhat
  rss <- sum((y1-chat)^2+lambda*alphas*(chat-yhat)^2)
  list(label=chat, deviance=rss)
}

aim2.split <- function(y, wt, x, parms, continuous)
  {
    n <- length(y[,1])
    y1 <- y[,1]
    yhat <- y[,2]
    alpha <- y[,3]
    lambda <- parms$lambda
    if (continuous)
      {
        if(is.null(lambda)) compute.lambda #Placeholder until I figure out how to compute lambda
        goodness <- direction <- double(n-1) #Allocate 0 vector
        y.cumsum <- cumsum(y1)
        y.left <- y.cumsum[-n]
        y.right <- y.cumsum[n]-y.left
        yhat.cumsum <- cumsum(yhat*alpha)
        yhat.left <- yhat.cumsum[-n]
        yhat.right <- yhat.cumsum[n]-yhat.left
        alpha.cumsum <- cumsum(alpha)
        alpha.left <- alpha.cumsum[-n]
        alpha.right <- alpha.cumsum[n]-alpha.left
        for(i in 1:(n-1))
          {
            zbar.left <- y.left[i]/i
            zbar.right <- y.right[i]/(n-i)
            zbarhat.left <- yhat.left[i]/alpha.left[i]
            zbarhat.right <- yhat.right[i]/alpha.right[i]
            alphabar.left <- alpha.left[i]/i
            alphabar.right <- alpha.right[i]/(n-i)
            r.left <- 1/(1+lambda*alphabar.left)
            r.right <- 1/(1+lambda*alphabar.right)
            chat.left <- r.left*zbar.left+(1-r.left)*zbarhat.left
            chat.right <- r.right*zbar.right+(1-r.right)*zbarhat.right

#            goodness[i] <- sum((y1-mean(y1))^2)
```

```
#                 - (sum((y1[1:i]-chat.left)^2 +
#                    lambda*alpha[1:i]*(yhat[1:i]-chat.left)^2) +
#                   sum((y1[(i+1):n]-chat.right)^2 +
#                  lambda*alpha[(i+1):n]*(yhat[(i+1):n]-chat.right)^2))
#           Do we need adjustment for missing values like in  vignette example?

            direction[i] <- sign(zbar.left-zbar.right)
            goodness.left <- sum((y1[1:i]-chat.left)^2 + lambda*alpha[1:i]*(yhat[1:i]-chat.left)^2)
            goodness.right <- sum((y1[(i+1):n]-chat.right)^2 +
                                    lambda*alpha[(i+1):n]*(yhat[(i+1):n]-chat.right)^2)
            tss <- sum((y1-mean(y1))^2)
            goodness[i] <- tss-goodness.left-goodness.right

          }
      } # this means we can only have x continuous - no categorical
#    goodness <- 1/goodness
    return(list(goodness=goodness, direction=direction))
  }

aim2.summary <- function(yval, dev, wt, ylevel, digits )
{
  paste(" mean=", format(signif(yval, digits)), ", MSE=" , format(signif(dev/wt, digits)), sep= '')
}

aim2.text <- function(yval, dev, wt, ylevel, digits, n, use.n )
{
  if(use.n) paste(formatg(yval,digits)," nn=", n,sep="")
  else paste(formatg(yval,digits))
}
```

## Example: Boston housing data

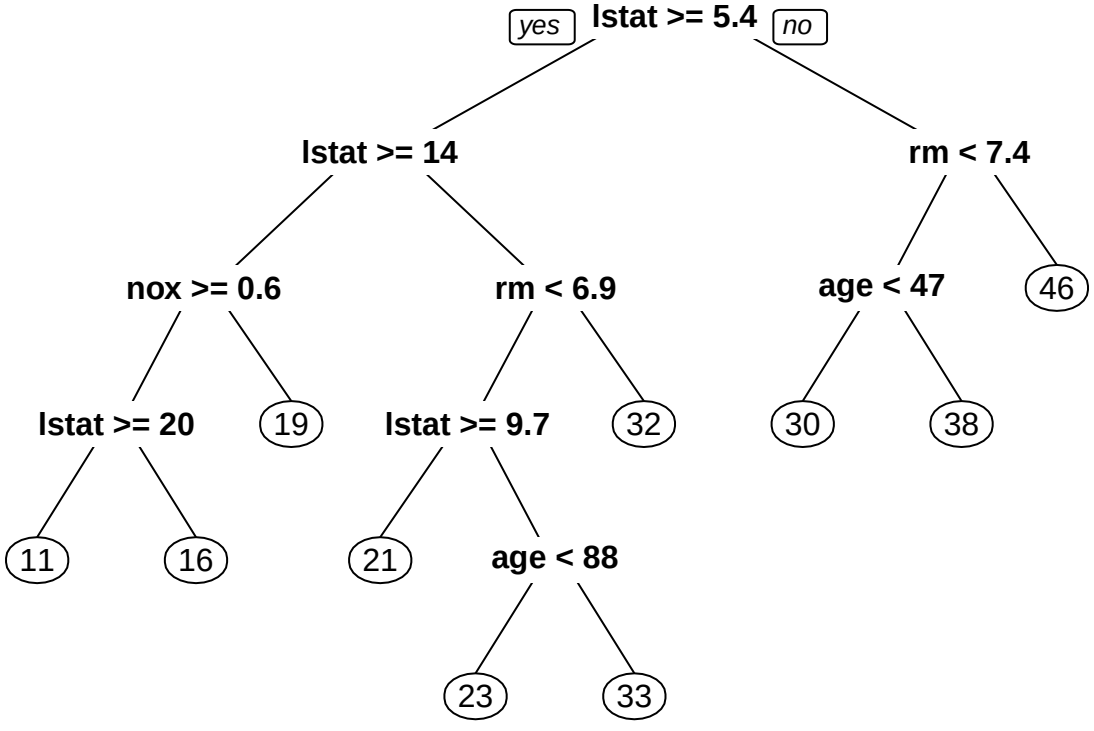
For an example, we have the Boston housing data. In the following block of text we read in the data, rename the columns, run the algorithm (via `aim2` function) and then plot the errors.

```
housing.data <- as.data.frame(matrix(scan("housing.data"),nrow=506,byrow=TRUE))
colnames(housing.data) <- c("crim","zn","indus","chas","nox","rm","age","dis","rad",
                            "tax","ptratiob","b","lstat","mdev")
temp <- aim2(dat=housing.data,nreps=1,n.grid=20,mult=2,seed=12345,outvar="mdev")

  plot(temp$lambdas,temp$Error.lambdas,xlab="lambda",ylab="Optimism corrected error",
       main="Optimism corrected error vs. lambda for housing data")
```




According to this plot, we would select the lambda equal to 3.6399528 is the one which minimizes the optimism corrected error. Using that lambda we get the following `rpart` tree:

```
## Warning: Unrecognized rpart object: treating as a numeric response model
```



# Appendix: Important background calculations

**Make sure learning/test is updated and that $\mathcal{M}$ is used for final model**

## Background for estimating $c_L$ and $c_R$

Let $\omega_i, i \geq 1$ be nonnegative weights, where at least one is positive. Let $\alpha_i, i \geq 1$ and $A_i, i \geq 1$ respectively be sequences of positive and real-valued constants. Let $Z_i, i \geq 1$ be a sequence of random variables. Finally let $\lambda > 0$ be given and consider the problem of minimizing

$$Q(c) = \sum_i \omega_i \left[ (Z_i - c)^2 + \lambda \alpha_i (c - A_i)^2 \right]$$

in $c$. If we differentiate $Q(c)$ with respect to $c$, set $Q'(c) = 0$, and solve for $c$, we obtain

$$c(\lambda) = \frac{\sum_i \omega_i Z_i}{\lambda \sum_i \omega_i \alpha_i + \sum_i \omega_i} + \frac{\lambda \sum_i \omega_i \alpha_i A_i}{\lambda \sum_i \omega_i \alpha_i + \sum_i \omega_i}.$$

Doing a bit of algebra,

$$c(\lambda) = r(\lambda) \frac{\sum_i \omega_i Z_i}{\sum_i \omega_i} + (1 - r(\lambda)) \frac{\sum_i \omega_i \alpha_i A_i}{\sum_i \omega_i \alpha_i}. \tag{16}$$

where

$$r(\lambda) = \frac{\sum_i \omega_i}{\lambda \sum_i \omega_i \alpha_i + \sum_i \omega_i} = \frac{1}{(1 + \lambda \bar{\alpha})}, \tag{17}$$

and

$$\bar{\alpha} = \frac{\sum_i \omega_i \alpha_i}{\sum_i \omega_i}. \tag{18}$$

Clearly, $r(\lambda) \in [0, 1]$ and so this is just a shrinkage estimator that balances the observed weighted average (which we'd get if $\lambda = 0$)

$$\frac{\sum_i \omega_i Z_i}{\sum_i \omega_i}$$

with the $\alpha-$modified weighted average of the $A$s

$$\frac{\sum_i \omega_i \alpha_i A_i}{\sum_i \omega_i \alpha_i}.$$

(which we'd get as $\lambda \to \infty$).

## Derivation of Within-Node Prediction Error

Now, let $\tilde{Z}$ be independent of $H = \{Z_i, i \geq 1\}$. We can define the conditional prediction error using $c(\lambda)$ as

$$CPE(\lambda) = E\left[ (\tilde{Z} - c(\lambda))^2 | H \right]$$

and the prediction error $PE(\lambda) = E_H \left[ CPE(\lambda) \right]$ (here, $E_H$ denotes the expectation wrt distribution of $H$). We would like to know what $\lambda$ minimizes $PE(\lambda)$. Note that $c(\lambda)$ is known given $H$ under the assumptions made at the beginning of the previous subsection.

We can write

$$(\tilde{Z} - c(\lambda))^2 = \tilde{Z}^2 - 2\tilde{Z}c(\lambda) + [c(\lambda)]^2.$$

Defining $\sigma_Z^2 = var(\tilde{Z})$ and $\mu_Z = E[\tilde{Z}]$ we have

$$CPE(\lambda) = \sigma_Z^2 + \mu_Z^2 - 2\mu_Z c(\lambda) + [c(\lambda)]^2.$$

Hence

$$PE(\lambda) = \sigma_Z^2 + \mu_Z^2 - 2\mu_Z E_H[c(\lambda)] + E_H[[c(\lambda)]^2].$$

Let $\mu_c(\lambda) = E_H[c(\lambda)]$ and $\sigma_c^2(\lambda) = var_H(c(\lambda))$ ; then, we can rewrite this last expression as

$$PE(\lambda) = \sigma_Z^2 + \mu_Z^2 - 2\mu_Z\mu_c(\lambda) + \sigma_c^2(\lambda) + \mu_c^2(\lambda).$$

Now, suppose $E[Z_i] = \delta$ for each $i$; then,

$$\mu_c(\lambda) = r(\lambda)\delta + K_1(1 - r(\lambda)).$$

for

$$K_1 = \frac{\sum_i \omega_i\alpha_i A_i}{\sum_i \omega_i\alpha_i}.$$

Similarly, if $var(Z_i) = \gamma$ for each $i$, then

$$var_c(\lambda) = r^2(\lambda)K_2\gamma$$

for

$$K_2 = \frac{\sum_i \omega_i^2}{[\sum_i \omega_i]^2}$$

As result we may write

$$PE(\lambda) = \sigma_Z^2 + \mu_Z^2 - 2\mu_Z[r(\lambda)\delta + K_1(1 - r(\lambda))] + r^2(\lambda)K_2\gamma + [r(\lambda)\delta + K_1(1 - r(\lambda))]^2$$

In the special case where $\delta = \mu_Z$ and $\gamma = \sigma_Z^2$, differentiating $PE(\lambda) = 0$ with respect to $\lambda$ and solving $PE'(\lambda) = 0$ gives

$$\lambda_0 = \frac{K_2\sigma_Z^2}{\bar{\alpha}(\mu_z - K_1)^2}, \tag{19}$$

where $\bar{\alpha}$ is given in (18).

To connect the notation of the Aim2b set-up and the notation of Section , let $\omega_i = I\{W_{1i} \in Q_v(j,s)\}$, $Z_i = Z_{1i}$ and $A_i = \hat{Z}_{1i}$.

## Alternative view of Strategy 2

.

Let $\omega_i = I\{W_{1i} \in Q_v(j,s)\}$, $Z_i = Z_{1i}$ and $A_i = A_{v,j,s}I\{W_{1i} \in Q_v(j,s)\}$ (i.e., $A_{v,j,s}$ doesn't depend on $i$ and thus is constant within node). Then, the formulas (16)-(18) of Section give

$$\widehat{c}_{v,j,s}(\lambda) = r_{v,j,s}(\lambda)\bar{Z}_v(j,s) + (1 - r_{v,j,s}(\lambda))\hat{\bar{A}}_{1v}(j,s),$$

where $r_{v,j,s}(\lambda) = 1/(1 + \lambda\bar{\alpha}_{v,j,s})$,

$$\hat{\bar{A}}_{1v}(j,s) = \{\sum_i I(W_{1i} \in Q_v(j,s))\alpha_i A_{v,j,s}\}/\{\sum_i I(W_{1i} \in Q_v(j,s))\alpha_i\} = A_{v,j,s}$$

and

$$\bar{\alpha}_{v,j,s} = n_{v,j,s}^{-1} \sum_i I(W_{1i} \in Q_v(j,s))\alpha_i$$

where $n_{v,j,s} = \sum_i I(W_{1i} \in Q_v(j,s))$.

Applying the results of Section  and assuming all expectation calculations are conditional on $W_{1i}, i \geq 1$ it can be shown that

$$K_2 = n_{v,j,s}^{-1}$$

and

$$K_1 = A_{v,j,s}.$$

Assuming that $E(Z_i) = \mu_Z$ and $var(Z_i) = \sigma_Z^2$ when $I(W_{1i} \in Q_v(j,s)) = 1$ (i.e., constant mean and variance within a node), the "best" within-node choice of $\lambda$ via (**??**) becomes

$$\lambda_{opt} = \frac{n_{v,j,s}^{-1}\sigma_Z^2}{\bar{\alpha}_{v,j,s}(\mu_z - A_{v,j,s})^2}.$$

Notice that selecting

$$A_{v,j,s} = \hat{\bar{Z}}_{1v}(j,s) = \{\sum_i I(W_{1i} \in Q_v(j,s))\alpha_i \hat{Z}_{1i}\}/\{\sum_i I(W_{1i} \in Q_v(j,s))\alpha_i\}$$

gives the same results as in the last section. Selecting instead

$$A_{v,j,s} = \hat{\bar{Z}}_{1v}(j,s) = \{\sum_i I(W_{1i} \in Q_v(j,s))\hat{Z}_{1i}\}/\{\sum_i I(W_{1i} \in Q_v(j,s))\}$$

gives an alternative shrinkage target. There are other choices as well.\

The point here is that Strategy 2 can be viewed as a procedure for shrinking the node-specific estimates towards some node-specific average predicted value.

# To-Do List

For the short term
1. Decide if want to use alternate shrinkage target or original (the initial lambda is chosen based on root node with the alternate shrinkage target)
2. Update code so last column of data is not assumed to be outcome
3. Currently - can only have x as continuous - in aim2.split option for having categorical variables has not been coded.