8: Tree-based regression

John H Maindonald

April 3, 2018

Ideas and issues illustrated by the graphs in this vignette

The fitting of a tree proceeds by making a succession of splits on the x-variable or variables. For tree-based regression, the splitting criterion is named `Anova`. (To explicitly request use of this criterion, specify `method="anova"` in the call to `rpart()`.)

1 Code for the Figures

```
fig8.1A <- function(){
  if(!exists('car90.rpart'))
  car90.rpart <- rpart(Mileage ~ tonsWt, data=Car90)
  plot(car90.rpart)
text(car90.rpart, xpd=TRUE, digits=3)
mtext(side=3, line=1.25, "A: Regression tree", adj=0)
}
```

```
fig8.1B <- function(){
if(!exists('car90.rpart'))
  car90.rpart <- rpart(Mileage ~ tonsWt, data=Car90)
plot(Mileage ~ tonsWt, data=Car90)
wt <- with(Car90, tonsWt)
hat <- predict(car90.rpart)
addhlines(wt, hat, lwd=2, col="gray")
mtext(side=3, line=1.25, "B: Predicted values from tree", adj=0)
}
```

```
fig8.2 <- function(){
BSS <- bssBYcut(tonsWt, Mileage, Car90)
with(BSS, plot(xOrd, bss, xlab="Cutpoint",
               ylab="Between groups sum of squares"))
```

```
abline(v=1.218, lty=2)
}
```

```
fig8.3A <- function(){
opar <- par(mar=c(4,4,2.6,1.6))
if(!exists('car90x.rpart'))
  car90x.rpart <- rpart(Mileage ~ tonsWt, data=Car90,
                        minbucket=5, minsplit=10,
                        cp=0.001)
plot(car90x.rpart, uniform=TRUE)
text(car90x.rpart, digits=3, xpd=TRUE)
mtext(side=3, line=0.75, "A: Decision tree", adj=0)
par(opar)
}
```

```
opar <- par(mar=c(4,4,2.6,1.6))
fig8.3B <- function(){
if(!exists('car90x.rpart'))
  car90x.rpart <- rpart(Mileage ~ tonsWt, data=Car90,
                        minbucket=5, minsplit=10,
                        cp=0.001)
plot(Mileage ~ tonsWt, data=Car90)
hat <- predict(car90x.rpart)
wt <- with(Car90, tonsWt)
addhlines(wt, hat, lwd=2, col="gray")
mtext(side=3, line=0.75, "B: Mileage vs tonsWt", adj=0)
par(opar)
}
```

```
fig8.4 <- function(){
if(!exists('car90x.rpart'))
  car90x.rpart <- rpart(Mileage ~ tonsWt, data=Car90,
                        minbucket=5, minsplit=10,
                        cp=0.001)
plotcp(car90x.rpart)
}
```

```
fig8.5 <- function(){
if(!exists('car90.rf'))
  car90.rf <- randomForest(Mileage ~ tonsWt,
                          data=Car90)
```

```
plot(Mileage ~ tonsWt, data=Car90, type="n")
with(Car90, points(Mileage ~ tonsWt, cex=0.8))
hat <- predict(car90.rf)
with(Car90, points(hat ~ tonsWt, pch="-"))
}
```

```
fig8.6 <- function(){
ran <- range(errsmat)
at <- round(ran+c(0.02,-0.02)*diff(ran),2)
lis <- list(limits=ran, at=at, labels=format(at, digits=2))
lims=list(lis,lis,lis,lis,lis,lis)
library(lattice)
splom(errsmat,
      pscales=lims,
      par.settings=simpleTheme(cex=0.75),
      col=adjustcolor("black", alpha=0.5),
      panel=function(x,y,...){lpoints(x,y,...)
      panel.abline(0,1,col="gray")}
)
}
```

2 Show the Figures

```
pkgs <- c("rpart","mgcv","randomForest","gamclass")
z <- sapply(pkgs, require, character.only=TRUE, warn.conflicts=FALSE)
if(any(!z)){
  notAvail <- paste(names(z)[!z], collapse=", ")
  print(paste("The following packages should be installed:", notAvail))
}
```

```
if(!exists('Car90'))
Car90 <- na.omit(car90[, c("Mileage","Weight")])
## Express weight in metric tonnes
Car90 <- within(Car90, tonsWt <- Weight/2240)
```

```
getmeuse <- function(){
if(require('sp', quietly=TRUE)){
data("meuse", package="sp", envir = environment())
meuse <- within(meuse, {levels(soil) <- c("1","2","2")
```

```
                        ffreq <- as.numeric(ffreq)
                        loglead <- log(lead)
                        })
invisible(meuse)
} else if(!exists("meuse"))
  print("Dataset 'meuse' was not found, get from package 'sp'")
}
```

```
cfRF <- function(nrep=50){
form1 <- ~ dist + elev + soil + ffreq
form3 <- ~ s(dist, k=3) + s(elev,k=3) + soil +ffreq
form3x <- ~ s(dist, k=3) + s(elev,k=3) + s(x, k=3) + soil+ffreq
form8x <- ~ s(dist, k=8) + s(elev,k=8) + s(x, k=8) + soil+ffreq
formlist <- list("Hybrid1"=form1, "Hybrid3"=form3,
                 "Hybrid3x"=form3x, "Hybrid8x"=form8x)
## ----rfgam-setup----
rfVars <- c("dist", "elev", "soil", "ffreq", "x", "y")
errsmat <- matrix(0, nrep, length(formlist)+2)
dimnames(errsmat)[[2]] <- c(names(formlist), "rfTest", "rfOOB")
n <- 95
for(i in 1:nrep){
sub <- sample(1:nrow(meuse), n)
meuseOut <- meuse[-sub,]
meuseIn <- meuse[sub,]
errsmat[i, ] <- gamRF(formlist=formlist, yvar="loglead",
                      rfVars=rfVars,
                      data=meuseIn, newdata=meuseOut,
                      printit=FALSE)
}
invisible(errsmat)
}
```

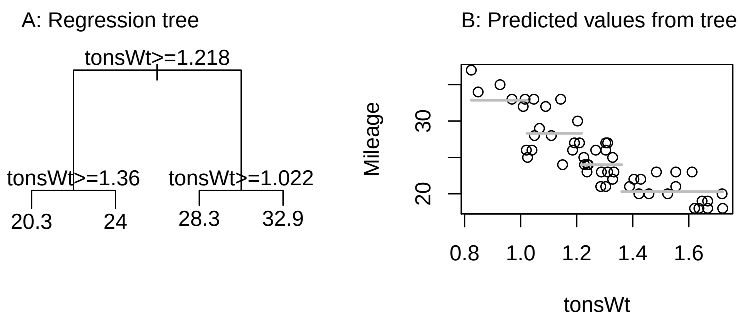

Figure 1: Regression tree for predicting `Mileage` given `Weight`. At each node, observations for which the criterion is satisfied take the branch to the left. Thus at the first node, `tonsWt`≥`1.218` chooses the branch to the left, while `tonsWt`<`1.218` chooses the branch to the right. Panel B plots `Mileage` versus `tonsWt`, with fitted values from the `rpart` model shown as horizontal grey lines.

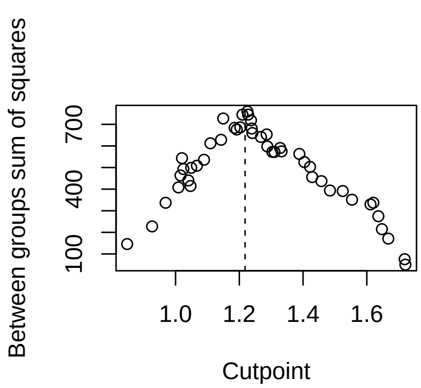

Figure 2: Between group sum of squares for `Mileage`, as a function of the value of `tonsWt` at which the split is made. The choice $c = 1.218$ maximizes the between groups sum of squares.

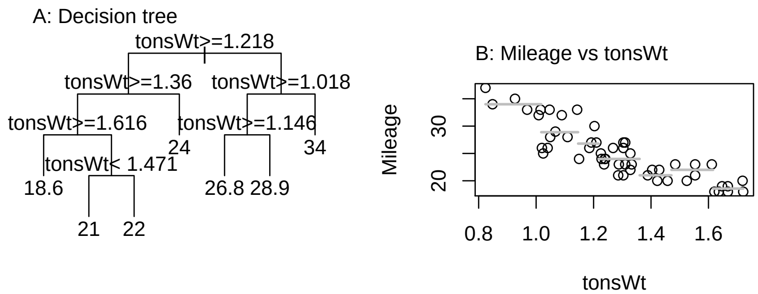

Figure 3: For the decision tree to which these panels relate, the minimum number at each terminal leaf (`minbucket`) has been reduced (from 10) to 5, the minimum number to allow further splitting (`minsplit`) has been reduced (from 20) to 10, and the complexity parameter has been reduced to `cp = 0.001` .

```
fig8.4()
```

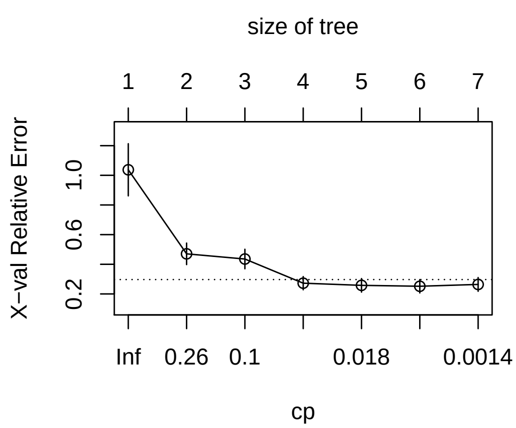

Figure 4: Change in cross-validated error rate, relative to baseline error, with successive splits. Because of the random element that arises from the cross-validation, the tree that is fitted and the pattern of change of cross-validated error will commonly change from one run to the next.

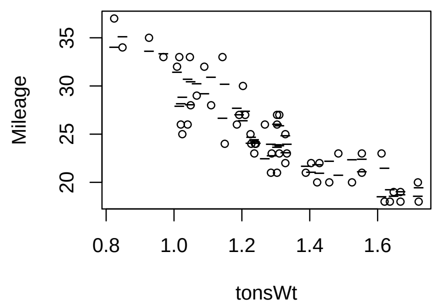

Figure 5: Plot of `Mileage` versus `tonsWt`, with fitted values from a `randomForest` regression shown as horizontal bars.

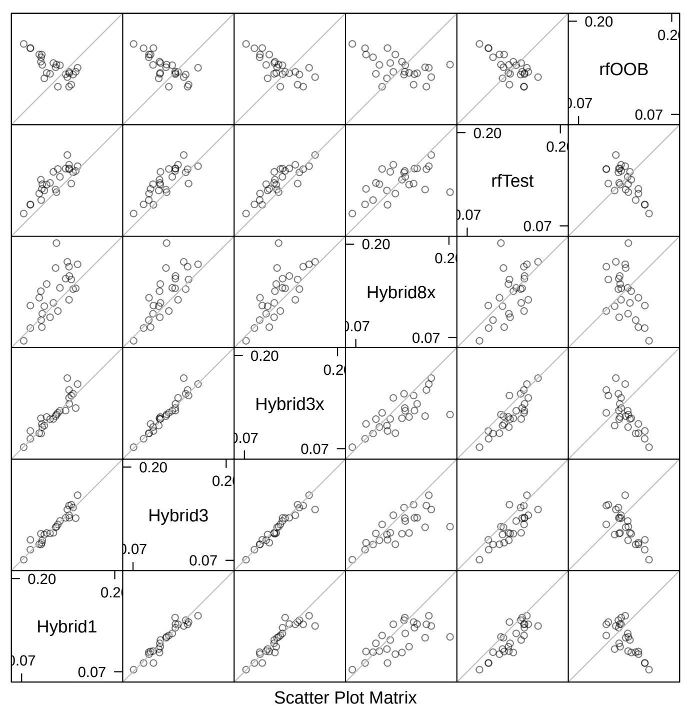

Figure 6: Scatterplot matrix of accuracies for the several models. Each panel shows the line $y = x$. The label rfOOB is out-of-bag accuracy for the 95 training set observations, while rfTest is test data accuracy, for a random forest model. Other results are test set accuracy from fitting a random forest model to residuals from a preliminary smooth. Labels are the name of the formula for the smooth. Random forest fits were from 25 bootstrap samples.