Regression Trees

Patrick Breheny

November 3

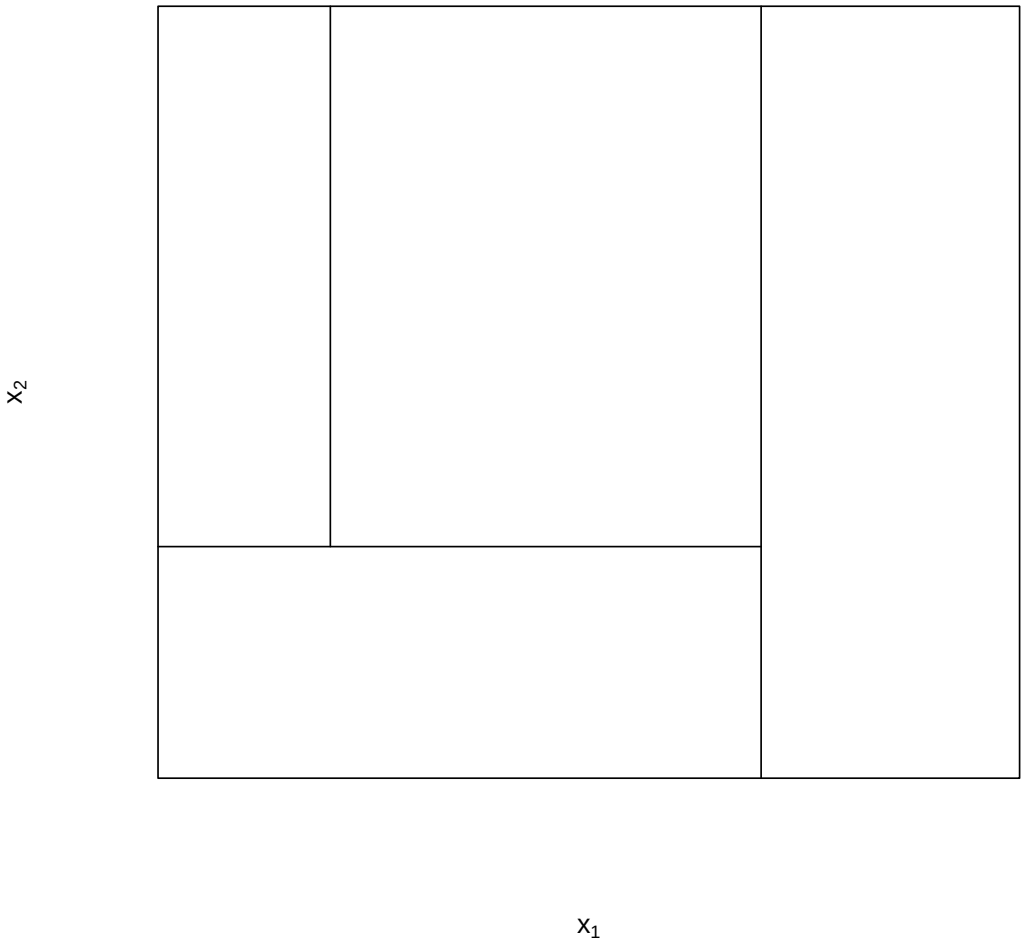
Introduction

- We've seen that local methods and splines both operate locally – either by using kernels to introduce differential weights or by using piecewise basis functions
- Either way, the kernels/basis functions were prespecified – *i.e.*, the basis functions are defined and weights given to observations regardless of whether they are needed to improve the fit or not
- Another possibility is to use the data to actively seek partitions which improve the fit as much as possible
- This is the main idea behind *tree-based methods*, which recursively partition the sample space into smaller and smaller rectangles

Recursive partitioning

- To see how this works, consider a linear regression problem with a continuous response y and two predictors x_1 and x_2
- We begin by splitting the space into two regions on the basis of a rule of the form $x_j \leq s$, and modeling the response using the mean of y in the two regions
- The optimal split (in terms of reducing the residual sum of squares) is found over all variables j and all possible split points s
- The process is then repeated in a recursive fashion for each of the two sub-regions

Partitioning illustration



The regression model

- This process continues until some stopping rule is applied
- For example, letting $\{R_m\}$ denote the collection of rectangular partitions, we might continue partitioning until $|R_m| = 10$
- The end result is a piecewise constant model over the partition $\{R_m\}$ of the form

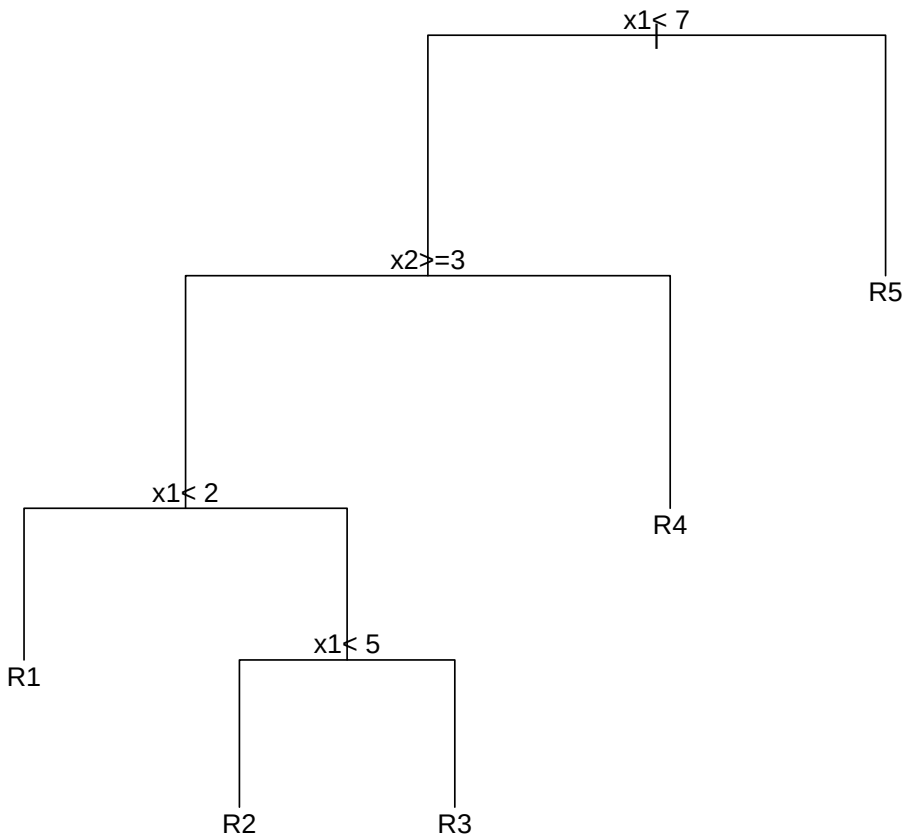
$$f(\mathbf{x}) = \sum_m c_m I(\mathbf{x} \in R_m)$$

where c_m is the constant term for the mth region (*i.e.*, the mean of y_i for those observations $\mathbf{x}_i \in R_m$)

Trees

- The same model can be neatly expressed in the form of a binary tree
- The regions $\{R_m\}$ are then referred to as the *terminal nodes* of the tree
- The non-terminal nodes are referred to as *interior nodes*
- The splits are variously referred to as "splits", "edges", or "branches"

Equivalent tree for example partition

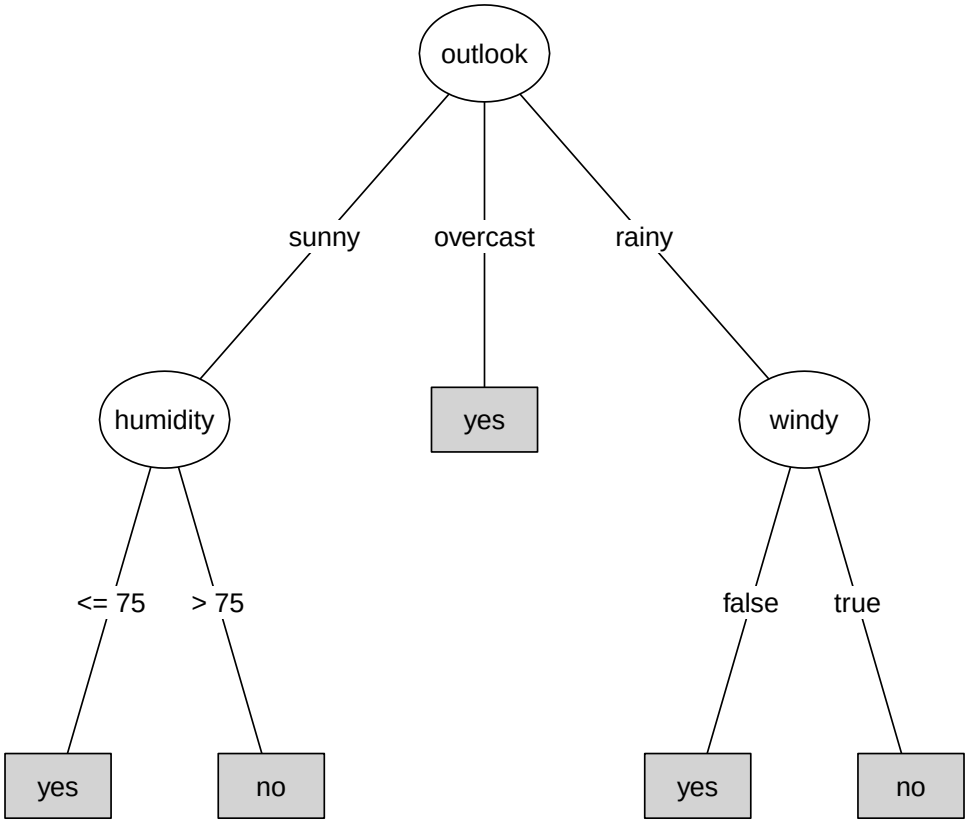


Trees and interpretability

- The ability to represent the model as a tree is the key to its interpretability and popularity
- With more than two explanatory variables, the earlier partition diagram becomes difficult to draw, but the tree representation can be extended to any dimension
- Trees are one of the most easily interpreted statistical methods: no understanding of statistics – or even mathematics – are required to follow them, and, to some extent, they mimic the way that human beings naturally think about things and make decisions

Artificial example

I know what the weather is like outside . . . should I play?



Algorithms vs. models

- As we will see, tree-based methods are not really statistical models – there is no distribution, no likelihood, no design matrix, none of the things we usually associate with modeling
- The thinking behind them is really more algorithmic, and treats the mechanism by which the data were generated as unknown and unknowable
- Admittedly, this is a bit foreign; however, in the words of Leo Breiman, one of the key pioneers of tree-based methods,

> *The statistical community has been committed to the almost exclusive use of data models. This commitment has led to irrelevant theory, questionable conclusions, and has kept statisticians from working on a large range of interesting current problems.*

Regression trees

- We now turn to some of the details involved in fitting trees, and begin with the case where our outcome is continuous (such trees are referred to as *regression trees*)
- First, note that if we adopt the least squares criterion as our objective, then our estimate for c_m is simply the average of the y_i's in that region:

$$\hat{c}_m = \frac{\sum_i y_i I(\mathbf{x} \in R_m)}{\sum_i I(\mathbf{x} \in R_m)}$$

- Our task is then to find the optimal splitting variable j and split point s that bring about the largest drop in the residual sum of squares

Regression tree algorithm

- For a given splitting variable j, this amounts to finding the value of s that minimizes

$$\sum_{i:x_j \leq s} (y_i - \hat{c}_1)^2 + \sum_{i:x_j > s} (y_i - \hat{c}_2)^2$$

- This may seem like a burdensome task, but if x_j has been sorted already, it can be done rather quickly
- Thus, we simply have to perform the above search for each variable j and then pick the best (j, s) pair for our split
- Having made this split, we then perform the whole process again on each of the two resulting regions, and so on

Categorical predictors

- In the preceding, we assumed that our predictors were continuous
- The exact same approach works for ordinal predictors
- For unordered categorical (*i.e.* nominal) predictors with q categories, there are $2^{q-1} - 1$ possible splits
- This actually makes things easier when q is small, but causes two problems when q is large:
  - The number of calculations grows prohibitive
  - The algorithm favors variables with a large number of possible splits, as the more choices we have, the better chance we can find one that seems to fit the data well

What size tree?

- How large should our tree be?
- Intuitively, a small tree might be too simple, while a large tree might overfit the data
- There are two main schools of thought on this matter:
  - The decision of whether to split or not should be based on a hypothesis test of whether the split significantly improves the fit or not
  - Tree size is a tuning parameter, and we can choose it using methods such as cross-validation

Hypothesis-testing approach

- The hypothesis-testing approach is straightforward (although the associated hypothesis tests may not be)
- Once you decide on the best split, perform a hypothesis test comparing the original model and the model that incorporates the new split
- If p is below some threshold, incorporate the new split and continue with the partitioning; if not, reject the split and terminate the algorithm

Pruning

- The upside of this approach, of course, is that you get p-values for each split, and they are guaranteed to be significant
- The downside is that a seemingly unimportant split might lead to a very important split later on
- The alternative is to "grow" a large tree, and then use a model-selection criterion to "prune" the tree back to its optimal size

Rules for growing trees

- Some common rules for when to stop growing a tree are:
  - When the number of terminal nodes exceeds some cutoff
  - When the number of observations in the terminal nodes reaches some cutoff
  - When the depth of the tree reaches a certain level
- Denote this tree, the largest tree under consideration, as T_0

Node impurity

- Now consider a subtree T that can be obtained by pruning T_0 – that is, by collapsing any number of its internal nodes
- Let $|T|$ denote the number of terminal nodes in tree T, and index those nodes with m, with node m representing region R_m
- We now define the *node impurity measure*:

$$Q_m(T) = \frac{1}{N_m} \sum_{i:x_i \in R_m} (y_i - \hat{c}_m)^2,$$

where N_m is the number of observations in node m

Cost-complexity pruning

- Finally, we define the *cost-complexity* criterion:

$$C_\alpha(T) = \sum_m N_m Q_m(T) + \alpha |T|$$

- The tuning parameter α behaves like the other regularization parameters we have seen, balancing stability (tree size) with goodness of fit
- For any given α, there is a tree T_α which minimizes the above criterion
- As the notation suggests, with $\alpha = 0$ we get T_0, the full tree
- α itself is usually chosen via cross-validation

Cotinine data

- To get a sense of how trees and their implementation in SAS and R work, we now turn to an example involving second hand smoke exposure in children
- Cotinine is a metabolite of nicotine, and is found in elevated levels in people exposed to second-hand smoke
- Measurement of cotinine requires lab tests, which cost time and money
- It is easier, of course, to simply ask parents about the extent of second hand smoke that their children are exposed to – but how accurate are their answers?

Cotinine data (cont'd)

To assess the correspondence (or lack thereof) between self-reported exposure and cotinine levels, the following variables were recorded:

- `SmokerTime`: Time spent with smokers (Daily/Intermittent/None)
- `TSHours`: Hours/day spent with a smoker
- `Nsmokers`: Number of smokers who live in the household
- `PPD`: Packs per day smoked by the household
- `PctOut`: Percentage of time spent smoking that is done outdoors
- `SHS`: Self-reported second-hand smoke exposure (None/Mild/Moderate/Heavy)

Tree packages in R

- In R, there are two primary packages one can use to fit tree-based models:
  - `rpart`, which is based on cost-complexity pruning
  - `party`, which is based on hypothesis test-based stopping
- Another relevant difference between the packages is that `party` selects its splits in a manner that alleviates the problem alluded to earlier, whereby explanatory variables with a large number of possible splits are more likely to be selected

Usage

- In `rpart`, the model-fitting function is `rpart`:

```
fit0 <- rpart(Cotinine~.,data=shs)
```

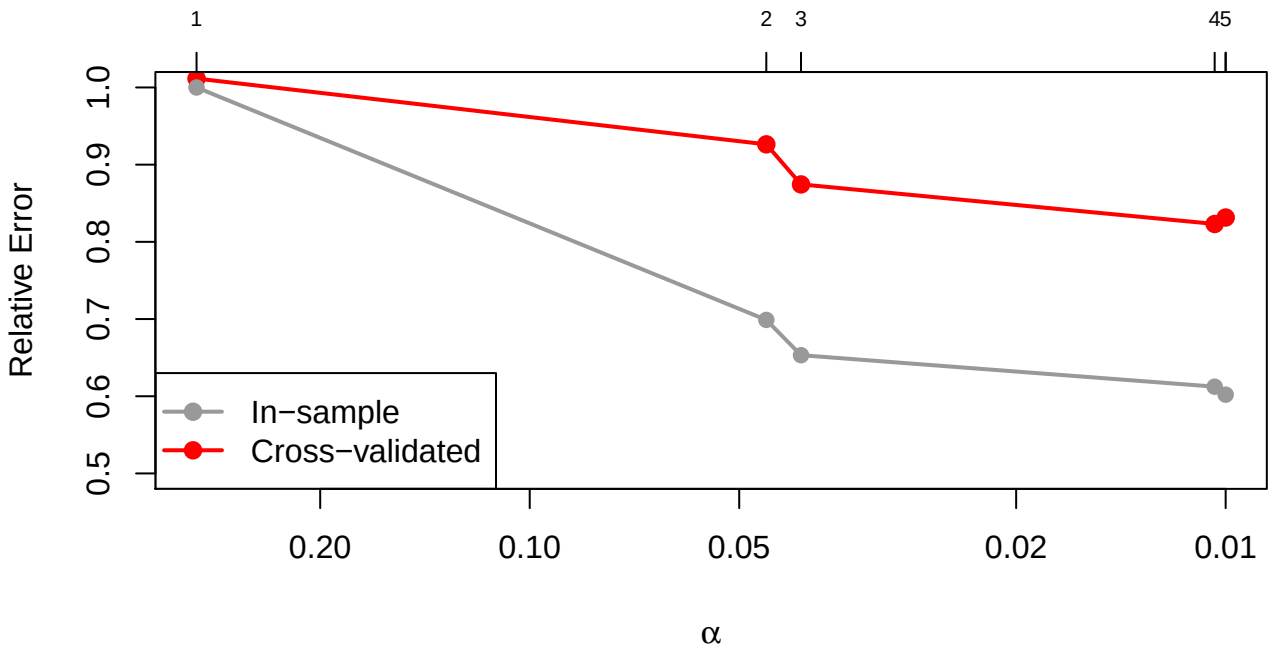
- In `party`, the model-fitting function is `ctree`:

```
fit <- ctree(Cotinine~.,data=shs)
```

Cost-complexity pruning

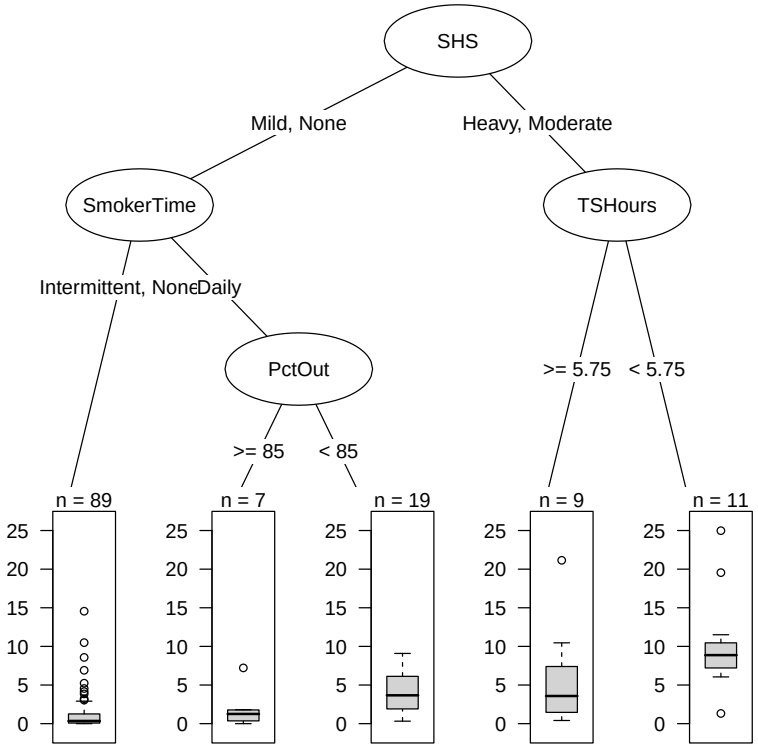
- In `party`, the algorithm stops automatically when further splits no longer significantly improve the fit
- In `rpart`, one still has to prune the tree after it has been grown
- Thankfully, `rpart` carries out cross-validation for you and stores the results in `fit$cptable`
- α is referred to as `cp`, and the cross-validation error is `xerror`

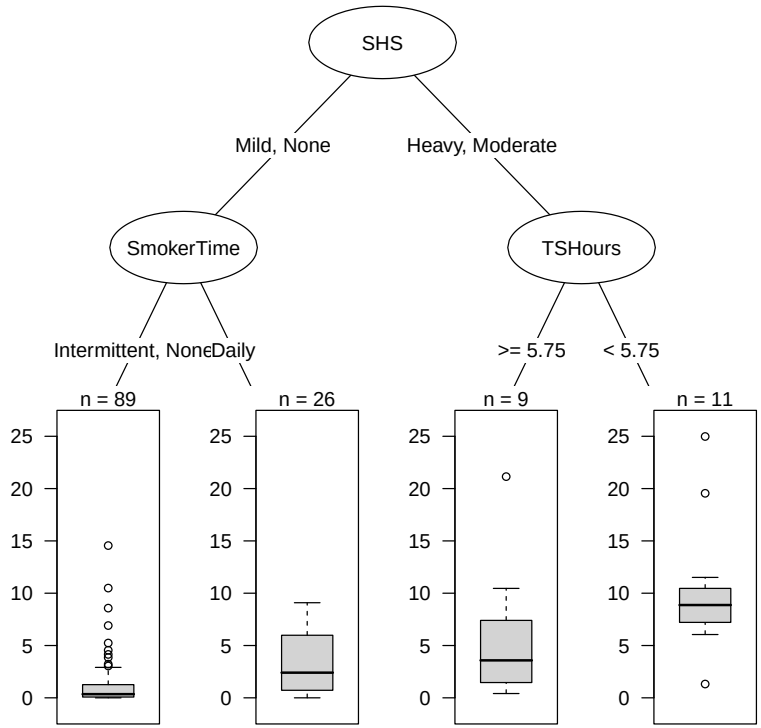
Cost-complexity pruning




```
alpha <- fit0$cptable[which.min(fit0$cptable[,"xerror"]),"CP"]
fit <- prune(fit0,alpha)
```

Original and pruned trees





Plotting trees

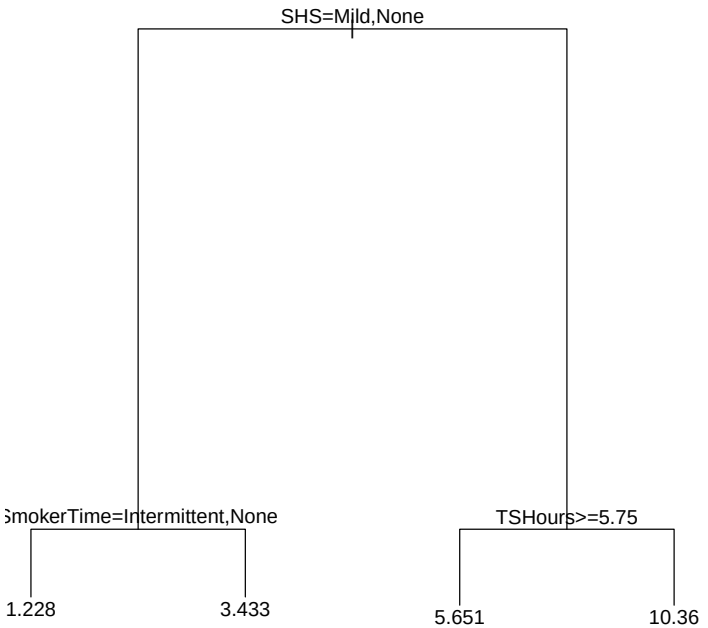
- `rpart` comes with its own plotting method:

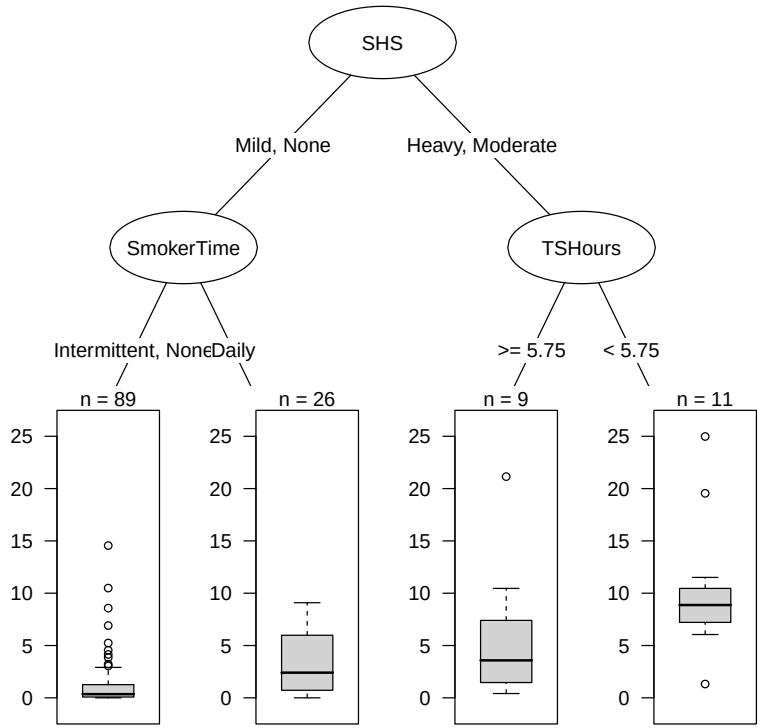
```
plot(fit)
text(fit)
```

- However, the end result is not particularly beautiful
- The plotting functions in `party` are much nicer; thankfully, you can use `party`'s tools to plot `rpart` objects using the package `partykit`:

```
require(partykit)
plot(as.party(fit))
```

Plotting trees (cont'd)





rpart vs. ctree