MSG500/MVE190
Linear Models - Lecture 14

Rebecka Jörnsten
Mathematical Statistics
University of Gothenburg/Chalmers University of Technology

December 11, 2012

1 Classification and Regression Trees

The linear models we have covered so far are ideal if the model is additive, and can be refined to include interactions. CART is a popular method that is ideal for interaction models where the outcome or response takes on a near constant value for a particular combination of x-values. Note, CART can also be generalized to models that are near linear for each x-value combination (called MARS) but we will work with the basic CART model here.

CART models (Classification and Regression Trees) are easy and intuitive way to summarize the data. CART is like a parlour game of ”Twenty Questions”. The questions you can ask are related to thresholds on the variables: e.g. ”Is the observed value of variable x_j less than or equal to 2, or greater than 2?”. This kind of question is called a ”split”. Depending on which side of the split an observation i falls, we will attribute a constant fitted value. There will be one fitted value for all the observations for which the answer to the above question is ”yes”, and similarly for all the observations for which the answer is ”no”. The gain of the split is measured in terms of RSS if our outcome is numerical, and deviance (-2*loglikelihood) or misclassification error if the outcome is categorical.

I will outline CART for the numerical response case. Here, before the split the RSS is

$$\sum_{i=1}^{n}(y_i - \bar{y})^2.$$

After the split, let's say n_1 observations answered ”yes”, and n_2 observations answered ”no”. The RSS has now been reduced to

$$\sum_{i:x_{ij}\leq 2}(y_i - \hat{\mu}_1)^2 + \sum_{i:x_{ij}>2}(y_i - \hat{\mu}_2)^2,$$

where

$$\hat{\mu}_1 = \sum_{i:x_{ij}\leq 2} y_i/n_1, \hat{\mu}_2 = \sum_{i:x_{ij}>2} y_i/n2.$$

For each branch we can now ask a new question to refine the predictions. Again, this leads to a reduction in RSS.

The combination of splits, e.g.

$$\{X_1 \leq 2\} \cap \{X_3 > 5\} \cap \{X_8 \leq 2.3\},$$

constitute a rectangle in, here, 3 dimensional space. This example rectangle corresponds to asking 3 questions. For each rectangle we calculate the mean value of the observations for which the answer to each question is ”yes”. That is the corresponding prediction.

The results of the splits are illustrated using a ”Tree”, where each split is a node and the branches correspond to the different answers. The length of the branches are drawn proportional to the reduction in RSS. At the bottom of the tree we have the so-called ”leaves” (a high-dimensional rectangle) for which we calculate the predicted value from the mean of the observations corresponding to ”yes” answers.

Let us now discuss how to choose the split. There are two choices to be made for each split: the variable j to split on, and where to split (2 in the above example). For each question you intend to ask, you only worry about the current question, not the future ones. Thus, you need to identify which of the total of p variables to ask about, and then perform a search for the optimal threshold. Optimality here refers to minimizing the RSS we obtain after the split. This is a greedy search since we only look at the optimal decision locally (the current question). Once we make a decision on a split we focus on the next question. We will ask one question for the ”yes” from the first question, and a potentially different question for the ”no”. We search for the optimal variable and threshold separately for each.

1.1 Validation and model selection

How many questions should we ask? If we ask questions until every observation has its own rectangle we are clearly over-fitting. On future data, it is not plausible that the same detailed questions will summarize the data well. To evaluate this we perform cross-validation. We cut the bottom branches of the tree (pruning) until the cross-validation error increases.

As we did in regression, we can use random splits to analyze the stability of the tree model. By counting the number of times a variable is picked to be part of the tree model, and how high up in the tree it occurs, your get a measure of variable importance.

1.2 CART for classification

CART can also be used for categorical outcome data. Instead of letting each leaf represent a mean value for $y_i : i$ in leaf, we apply a majority voting scheme. If the majority of the observations in a leaf correspond to category A, we vote A.

1.3 More about CART

There are extensions of CART called MARS (for linear models in each rectangle) and subsampling of variables for each split (Random Forests). Random Forests and the tree models averaged over random splits (called Bagging) are among the best off-the-shelf classifiers. You can learn more about these methods in the Multivariate class.

2 Demo 14

I will analyze a data set consisting of wines from two different regions in Australia. For each wine we have data on color, hue, alcohol content, etc. I will use the package `tree()`, mainly because it's easier to access which variables are used in the model. The `rpart()` package is the more commonly used CART program, with better validation methods included. Also, the demos use CART for 0/1 response data, but you can just as easily use it for numerical response. The only thing below you have to change is the prediction and the collection of performance statistics since it is set to compute misclassification errors below (replace it with pMSE everywhere).

```
> library(tree)
> wine <- read.table("wine2.txt", header = T)
> tree1 <- tree(as.factor(class) ~ alcohol + malicacid + alcalinity +
+     phenols + flavanoids + proanthocyanins + colorintensity +
+     hue, data = wine)

> par(mfrow = c(1, 1))
> plot(tree1)
> text(tree1)
```

Here is the same code for the `rpart()` package:

```
> library(rpart)
> wine <- read.table("wine2.txt", header = T)
> tree2 <- rpart(as.factor(class) ~ alcohol + malicacid + alcalinity +
```

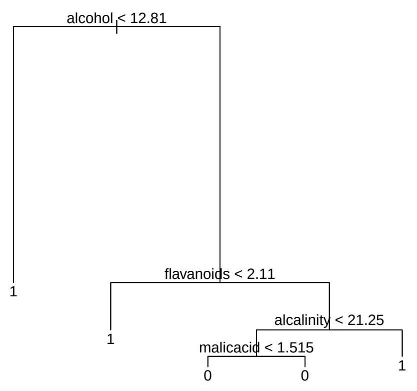

Figure 1: (tree) CART for the wine data.

```
+     phenols + flavanoids + proanthocyanins + colorintensity +
+     hue, data = wine)

> par(mfrow = c(1, 1))
> plot(tree2)
> text(tree2)
```

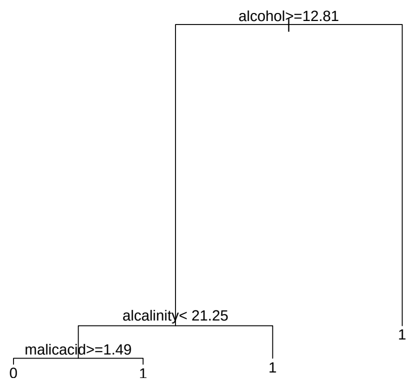

Figure 2: (rpart) CART for the wine data.

```
> cvtree1 <- cv.tree(tree1, K = 10)
> plot(cvtree1)
> print(cvtree1)

$size
[1] 5 4 3 2 1

$dev
[1]  47.78051  47.02670  53.27639  55.07763 181.10916
```

```
$k
[1]       -Inf   5.09952  12.96154  23.35838 125.95541

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
```

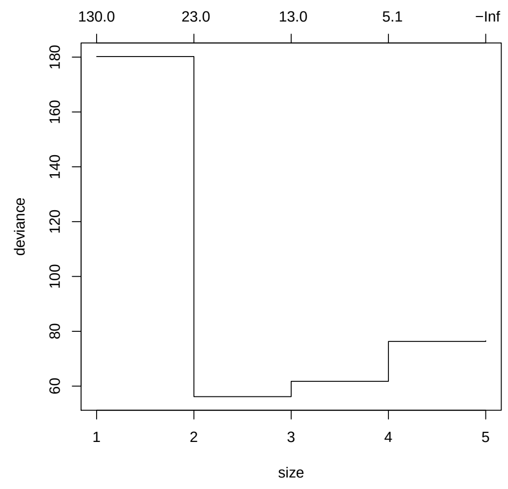

Figure 3: CV deviance for tree (CART)

In Figure 3 I depict the cross-validated deviance. Pick the tree with the smallest deviance.

```
> mim <- min(cvtree1$dev)
> sizesel <- max(2, max(cvtree1$size[cvtree1$dev == mim]))
> ptree1 <- prune.tree(tree1, best = sizesel)
> plot(ptree1)
> text(ptree1)
```

In Figure 4 the pruned tree is shown. I can use this for prediction:

```
> predtree <- predict.tree(ptree1, newdata = wine, type = "class")
> print(table(wine$class, predtree))

   predtree
     0  1
  0 59  0
  1  9 62
```

Note, this is the training error.

Using `rpart()` the validation is done using the `plotcp` command.

```
> plotcp(tree2)
```

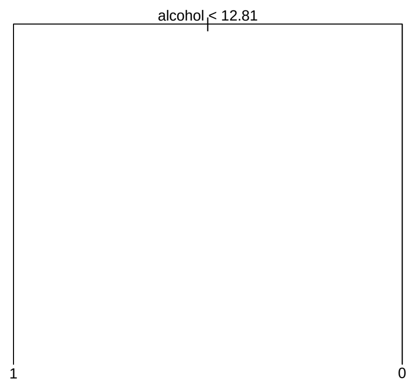

Figure 4: Pruned tree

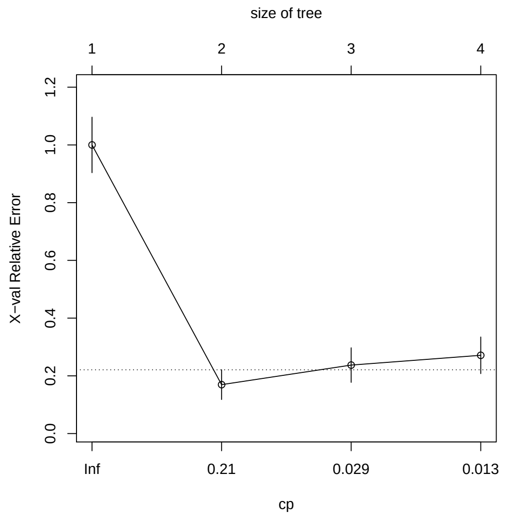

Figure 5: rpart validation plot. Pick the smallest tree such that the mean relative error is below the min error + its standard error (horizontal dashed line).

Returning to tree() and its generated model, let us try repeated splitting of the data into training and testing to see how the model performs.

```
> B <- 100
> Nbrleaves <- matrix(0, B, 1)
> UsedMatrix <- matrix(0, B, 8)
> UsedFirst <- matrix(0, B, 8)
> MSEMatrix <- matrix(0, B, 2)
> for (bb in (1:B)) {
+     iie <- sample(seq(1, dim(wine)[1]), 50)
+     trdata <- wine[-iie, ]
+     tedata <- wine[iie, ]
+     tree1 <- tree(as.factor(class) ~ alcohol + malicacid + alcalinity +
+         phenols + flavanoids + proanthocyanins + colorintensity +
+         hue, data = trdata)
+     cvtree1 <- cv.tree(tree1, K = 10)
+     mim <- min(cvtree1$dev)
+     sizesel <- max(2, max(cvtree1$size[cvtree1$dev == mim]))
+     ptree1 <- prune.tree(tree1, best = sizesel)
+     Nbrleaves[bb] <- length(unique(ptree1$where))
+     predtree <- predict.tree(ptree1, newdata = tedata, type = "class")
+     rlabels <- tedata[, 1]
+     MSEMatrix[bb, 1] <- sum(rlabels != predtree)/length(predtree)
+     predtree <- predict(tree1, newdata = tedata, type = "class")
+     MSEMatrix[bb, 2] <- sum(rlabels != predtree)/length(predtree)
+     UsedMatrix[bb, as.real(summary(ptree1)$used) - 1] <- UsedMatrix[bb,
+         as.real(summary(ptree1)$used) - 1] + 1
+     UsedFirst[bb, as.real(summary(ptree1)$used)[1] - 1] <- UsedFirst[bb,
+         as.real(summary(ptree1)$used)[1] - 1] + 1
+ }
```

The above code stores the used variables (both in terms of overall appearance, and also if they get picked as the first variable used in the tree).

```
> boxplot(MSEMatrix[, 1] - MSEMatrix[, 2], main = "PE-CV - PE-full")
> abline(h = 0)
> print(c(mean(MSEMatrix[, 1]), mean(MSEMatrix[, 2])))

[1] 0.0898 0.0966
```

In Figure 6 I show the prediction error difference using a pruned tree and full tree. Above I also summarize the mean misclassification error rate for the pruned tree models and the full tree models.

```
> sizetrees <- apply(UsedMatrix, 1, sum)
> hist(sizetrees, main = "Size trees")
> print(c("NbrLeaves", fivenum(Nbrleaves)))

[1] "NbrLeaves" "2"         "2"         "2"         "3"         "5"

> modtab <- apply(UsedMatrix, 2, sum)/B
> print(cbind(names(wine)[-1], modtab))

                       modtab
[1,] "alcohol"         "0.98"
[2,] "malicacid"       "0.16"
[3,] "alcalinity"      "0.11"
[4,] "phenols"         "0.04"
[5,] "flavanoids"      "0.13"
[6,] "proanthocyanins" "0"
[7,] "colorintensity"  "0.18"
[8,] "hue"             "0"
```

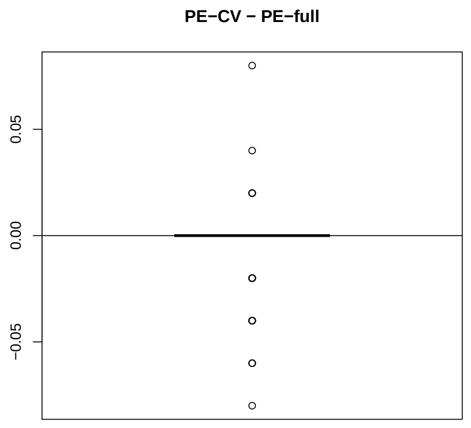

Figure 6: Prediction error (Pruned tree - full tree)

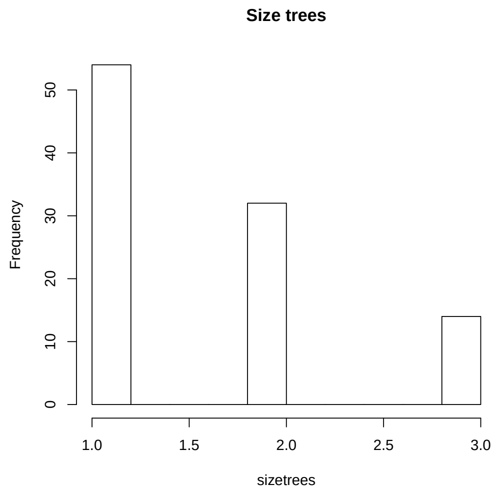

Figure 7: Size of pruned trees

```
> modfirst <- apply(UsedFirst, 2, sum)/B
> print(cbind(names(wine)[-1], modfirst))

                         modfirst
[1,] "alcohol"           "0.96"
[2,] "malicacid"         "0"
[3,] "alcalinity"        "0"
[4,] "phenols"           "0"
[5,] "flavanoids"        "0"
[6,] "proanthocyanins"   "0"
[7,] "colorintensity"    "0.04"
[8,] "hue"               "0"
```

Let's also try CART on the heart disease data.

```
> library(tree)
> SA <- read.table("SA.dat", header = T)

> tree1 <- tree(as.factor(chd) ~ log(age) + sbp + adiposity + obesity +
+     typea + alcohol + as.factor(alcind) + tobacco + as.factor(tobind) +
+     as.factor(famhist) + log(ldl), data = SA, minsize = 5)
> par(mfrow = c(1, 1))
> plot(tree1)
> text(tree1)
```

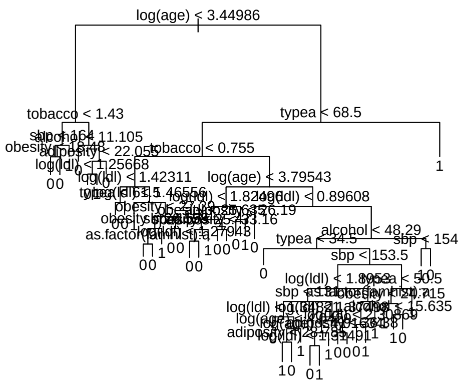

Figure 8: Heart disesase - full tree

```
> cvtree1 <- cv.tree(tree1, K = 10)
> plot(cvtree1)
> print(cvtree1)

$size
 [1] 43 42 41 39 38 37 34 32 31 29 28 26 24 21 20 19 18 14 13  7  6  4  3  2  1

$dev
 [1] 862.9800 804.7202 804.7202 795.7206 709.9599 706.5886 695.7419 694.0227
 [9] 672.5868 637.5689 608.9491 610.9610 600.4204 604.9403 606.8965 582.4419
[17] 585.7963 573.8722 508.7562 508.7562 470.9353 433.7639 432.4491 421.1345
[25] 422.5035

$k
 [1]      -Inf  4.556689  4.575312  4.708744  5.178277  5.268063  5.334298
 [8]  5.438701  5.487169  5.876211  5.973693  6.212506  6.250428  6.350974
[15]  6.401576  6.578206  6.734171  6.791491  7.343314  7.360236  9.718246
[22] 10.983953 12.883866 14.535413 41.383382

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
```

```
> mim <- min(cvtree1$dev)
> sizesel <- max(2, max(cvtree1$size[cvtree1$dev == mim]))
> ptree1 <- prune.tree(tree1, best = sizesel)
> plot(ptree1)
> text(ptree1)
```

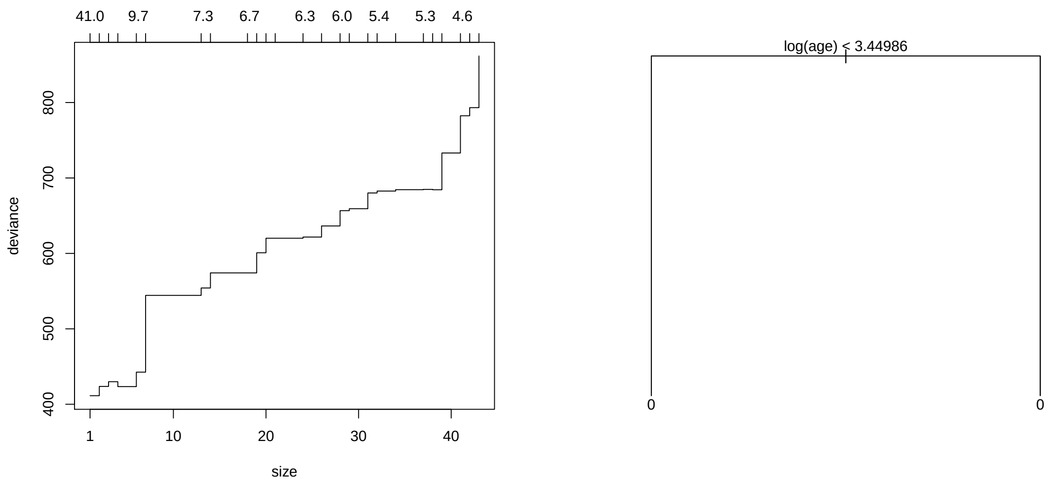

Figure 9: Heart disesase - CV deviance and pruned tree.

```
> predtree <- predict.tree(ptree1, newdata = SA, type = "class")
> print(table(SA$chd, predtree))

   predtree
      0   1
  0 205   0
  1 107   0
```

The error rate is not so good for the heart disease data (which we know from trying logistic regression).

Let's try random splits of the data to check model stability and variable importance, as well as prediction performance.

```
> B <- 100
> NbrLeaves <- matrix(0, B, 1)
> UsedMatrix <- matrix(0, B, 11)
> UsedFirst <- matrix(0, B, 11)
> MSEMatrix <- matrix(0, B, 2)
> for (bb in (1:B)) {
+     iie <- sample(seq(1, dim(SA)[1]), 50)
+     trdata <- SA[-iie, ]
+     tedata <- SA[iie, ]
+     tree1 <- tree(as.factor(chd) ~ log(age) + sbp + adiposity +
+         obesity + typea + alcohol + as.factor(alcind) + tobacco +
+         as.factor(tobind) + as.factor(famhist) + log(ldl), data = trdata,
+         minsize = 2)
+     cvtree1 <- cv.tree(tree1, K = 10)
+     mim <- min(cvtree1$dev)
+     sizesel <- max(2, max(cvtree1$size[cvtree1$dev == mim]))
+     ptree1 <- prune.tree(tree1, best = sizesel)
+     NbrLeaves[bb] <- length(unique(ptree1$where))
+     predtree <- predict.tree(ptree1, newdata = tedata, type = "class")
+     rlabels <- tedata[, 10]
+     MSEMatrix[bb, 1] <- sum(rlabels != predtree)/length(predtree)
+     predtree <- predict(tree1, newdata = tedata, type = "class")
+     MSEMatrix[bb, 2] <- sum(rlabels != predtree)/length(predtree)
+     UsedMatrix[bb, as.real(summary(ptree1)$used) - 1] <- UsedMatrix[bb,
+         as.real(summary(ptree1)$used) - 1] + 1
+     UsedFirst[bb, as.real(summary(ptree1)$used)[1] - 1] <- UsedFirst[bb,
+         as.real(summary(ptree1)$used)[1] - 1] + 1
+ }
```

The mean prediction error and summmary of the selected tree sizes are

```
> print(mean(MSEMatrix[, 1]))

[1] 0.3576

> sizetrees <- apply(UsedMatrix, 1, sum)
> print(fivenum(sizetrees))

[1] 1 1 1 2 5

> print(fivenum(NbrLeaves))

[1] 2 2 2 3 6

> modtab <- apply(UsedMatrix, 2, sum)/B
> print(cbind(names(SA)[-10], modtab))

                  modtab
 [1,] "age"       "0.73"
 [2,] "sbp"       "0.01"
 [3,] "adiposity" "0"
 [4,] "obesity"   "0.01"
 [5,] "typea"     "0.22"
 [6,] "alcohol"   "0"
 [7,] "alcind"    "0"
 [8,] "tobacco"   "0.44"
 [9,] "tobind"    "0"
[10,] "famhist"   "0.02"
[11,] "ldl"       "0.16"

> modfirst <- apply(UsedFirst, 2, sum)/B
> print(cbind(names(SA)[-10], modfirst))

                  modfirst
 [1,] "age"       "0.67"
 [2,] "sbp"       "0"
 [3,] "adiposity" "0"
 [4,] "obesity"   "0"
 [5,] "typea"     "0"
 [6,] "alcohol"   "0"
 [7,] "alcind"    "0"
 [8,] "tobacco"   "0.33"
 [9,] "tobind"    "0"
[10,] "famhist"   "0"
[11,] "ldl"       "0"
```

Let's try using CART to predict cholesterol.

```
> tree1 <- tree(ldl ~ log(age) + sbp + adiposity + obesity + typea +
+     alcohol + alcind + tobacco + tobind + as.factor(chd) + as.factor(famhist),
+     data = SA, minsize = 5)
> par(mfrow = c(1, 1))
> plot(tree1)
> text(tree1)
```

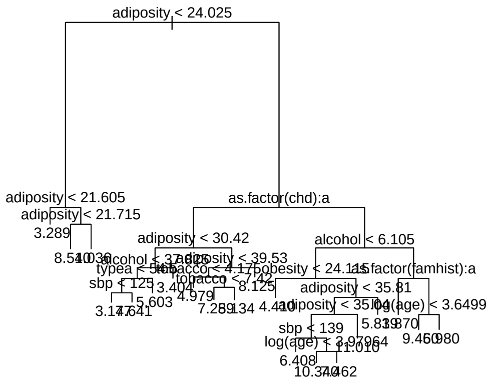

Figure 10: Heart disesase - predicting ldl - full tree

```
> cvtree1 <- cv.tree(tree1, K = 10)
> plot(cvtree1)
> print(cvtree1)

$size
 [1] 20 19 16 15 13 12 11 10  7  5  3  2  1

$dev
 [1] 1215.506 1192.125 1174.227 1144.154 1084.992 1074.909 1076.300 1093.349
 [9] 1093.349 1112.158 1109.147 1116.858 1271.342

$k
 [1]      -Inf  13.86829  15.87540  16.56961  18.73548  20.05683  22.42151
 [8]  29.72475  29.77289  30.93403  49.81390  60.39189 277.02590

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"

> mim <- min(cvtree1$dev)
> sizesel <- max(2, max(cvtree1$size[cvtree1$dev == mim]))
> ptree1 <- prune.tree(tree1, best = sizesel)
> plot(ptree1)
> text(ptree1)

> predtree <- predict.tree(ptree1, newdata = SA)
> print(sum((SA$ldl - predtree)^2)/length(predtree))
```

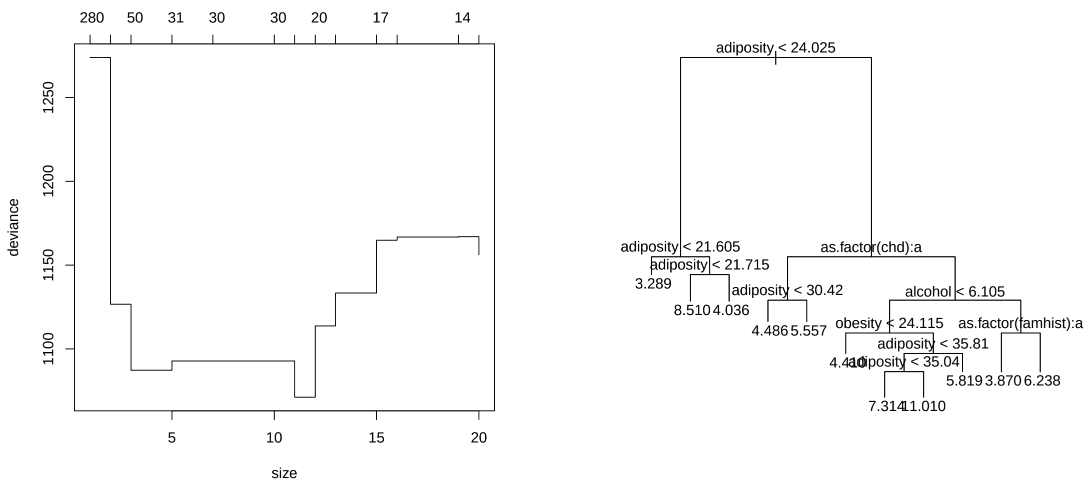

Figure 11: Heart disesase - ldl outcome - CV deviance and pruned tree.

```
[1] 2.07075
```

Let's try random splits of the data to check model stability and variable importance, as well as prediction performance.

```
> B <- 100
> NbrLeaves <- matrix(0, B, 1)
> UsedMatrix <- matrix(0, B, 11)
> UsedFirst <- matrix(0, B, 11)
> MSEMatrix <- matrix(0, B, 2)
> for (bb in (1:B)) {
+     iie <- sample(seq(1, dim(SA)[1]), 50)
+     trdata <- SA[-iie, ]
+     tedata <- SA[iie, ]
+     tree1 <- tree(ldl ~ log(age) + sbp + adiposity + obesity +
+         typea + alcohol + as.factor(alcind) + tobacco + as.factor(tobind) +
+         as.factor(chd) + as.factor(famhist), data = trdata, minsize = 2)
+     cvtree1 <- cv.tree(tree1, K = 10)
+     mim <- min(cvtree1$dev)
+     sizesel <- max(2, max(cvtree1$size[cvtree1$dev == mim]))
+     ptree1 <- prune.tree(tree1, best = sizesel)
+     NbrLeaves[bb] <- length(unique(ptree1$where))
+     predtree <- predict.tree(ptree1, newdata = tedata)
+     rlabels <- tedata[, 12]
+     MSEMatrix[bb, 1] <- sum((rlabels - predtree)^2)/length(predtree)
+     predtree <- predict(tree1, newdata = tedata)
+     MSEMatrix[bb, 2] <- sum((rlabels - predtree)^2)/length(predtree)
+     UsedMatrix[bb, as.real(summary(ptree1)$used) - 1] <- UsedMatrix[bb,
+         as.real(summary(ptree1)$used) - 1] + 1
+     UsedFirst[bb, as.real(summary(ptree1)$used)[1] - 1] <- UsedFirst[bb,
+         as.real(summary(ptree1)$used)[1] - 1] + 1
+ }
```

The mean prediction error and summmary of the selected tree sizes are

```
> print(c(mean(MSEMatrix[, 1]), mean(MSEMatrix[, 2])))
```

```
[1] 3.714802 4.288631
```

```
> sizetrees <- apply(UsedMatrix, 1, sum)
> print(fivenum(sizetrees))

[1] 1 1 3 5 9

> print(fivenum(NbrLeaves))

[1]  2  3  5  8 19

> modtab <- apply(UsedMatrix, 2, sum)/B
> print(cbind(names(SA)[-12], modtab))

                  modtab
 [1,] "age"       "0.36"
 [2,] "sbp"       "0.16"
 [3,] "adiposity" "0.94"
 [4,] "obesity"   "0.35"
 [5,] "typea"     "0.2"
 [6,] "alcohol"   "0.58"
 [7,] "alcind"    "0"
 [8,] "tobacco"   "0.17"
 [9,] "tobind"    "0.01"
[10,] "chd"       "0.49"
[11,] "famhist"   "0.2"

> modfirst <- apply(UsedFirst, 2, sum)/B
> print(cbind(names(SA)[-12], modfirst))

                  modfirst
 [1,] "age"       "0"
 [2,] "sbp"       "0"
 [3,] "adiposity" "0.86"
 [4,] "obesity"   "0.14"
 [5,] "typea"     "0"
 [6,] "alcohol"   "0"
 [7,] "alcind"    "0"
 [8,] "tobacco"   "0"
 [9,] "tobind"    "0"
[10,] "chd"       "0"
[11,] "famhist"   "0"
```