A Handbook of Statistical Analyses Using R — 3rd Edition

Torsten Hothorn and Brian S. Everitt

CHAPTER 9

Recursive Partitioning: Predicting Body Fat, Glaucoma Diagnosis, and Happiness in China

9.1 Introduction

9.2 Recursive Partitioning

9.3 Analysis Using R

9.3.1 Predicting Body Fat Content

The `rpart` function from **rpart** can be used to grow a regression tree. The response variable and the covariates are defined by a model formula in the same way as for `lm`, say. By default, a large initial tree is grown, we restrict the number of observations required to establish a potential binary split to at least ten:

```
R> library("rpart")
R> data("bodyfat", package = "TH.data")
R> bodyfat_rpart <- rpart(DEXfat ~ age + waistcirc + hipcirc +
+       elbowbreadth + kneebreadth, data = bodyfat,
+       control = rpart.control(minsplit = 10))
```

A `print` method for *rpart* objects is available; however, a graphical representation (here utilizing functionality offered from package **partykit**, Hothorn and Zeileis, 2014) shown in Figure 9.1 is more convenient. Observations that satisfy the condition shown for each node go to the left and observations that don't are an element of the right branch in each node. As expected, higher values for waist and hip circumferences and wider knees correspond to higher values of body fat content. The rightmost terminal node consists of only three rather extreme observations.

To determine if the tree is appropriate or if some of the branches need to be subjected to pruning we can use the `cptable` element of the *rpart* object:

```
R> print(bodyfat_rpart$cptable)
```

```
      CP nsplit rel error xerror   xstd
1 0.6629      0    1.0000  1.027 0.1684
2 0.0938      1    0.3371  0.427 0.0943
3 0.0770      2    0.2433  0.445 0.0869
4 0.0451      3    0.1663  0.354 0.0696
5 0.0184      4    0.1212  0.264 0.0597
```

```
R> library("partykit")
R> plot(as.party(bodyfat_rpart), tp_args = list(id = FALSE))
```

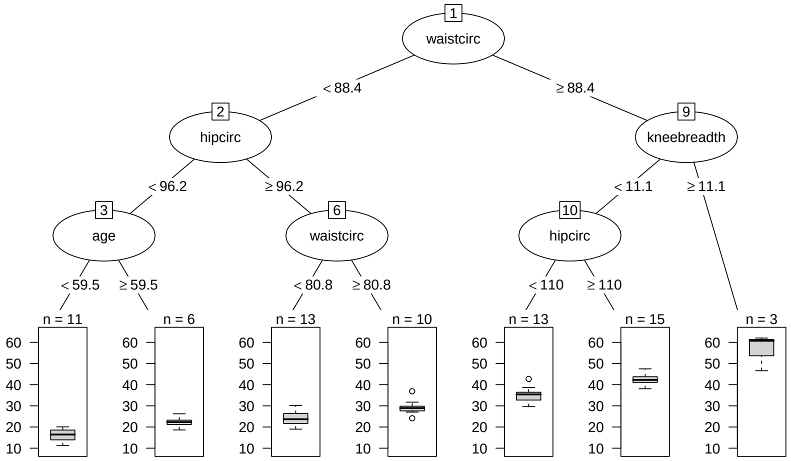

Figure 9.1 Initial tree for the body fat data with the distribution of body fat in terminal nodes visualized via boxplots.

```
6 0.0182       5     0.1028  0.286 0.0622
7 0.0100       6     0.0846  0.279 0.0624
R> opt <- which.min(bodyfat_rpart$cptable[,"xerror"])
```

The `xerror` column contains estimates of cross-validated prediction error for different numbers of splits (`nsplit`). The best tree has four splits. Now we can prune back the large initial tree using

```
R> cp <- bodyfat_rpart$cptable[opt, "CP"]
R> bodyfat_prune <- prune(bodyfat_rpart, cp = cp)
```

The result is shown in Figure 9.2. Note that the inner nodes three and six have been removed from the tree. Still, the rightmost terminal node might give very unreliable extreme predictions.

Given this model, one can predict the (unknown, in real circumstances) body fat content based on the covariate measurements. Here, using the known values of the response variable, we compare the model predictions with the actually measured body fat as shown in Figure 9.3. The three observations with large body fat measurements in the rightmost terminal node can be identified easily.

9.3.2 Glaucoma Diagnosis

```
R> data("GlaucomaM", package = "TH.data")
R> glaucoma_rpart <- rpart(Class ~ ., data = GlaucomaM,
R> plot(as.party(bodyfat_prune), tp_args = list(id = FALSE))
```

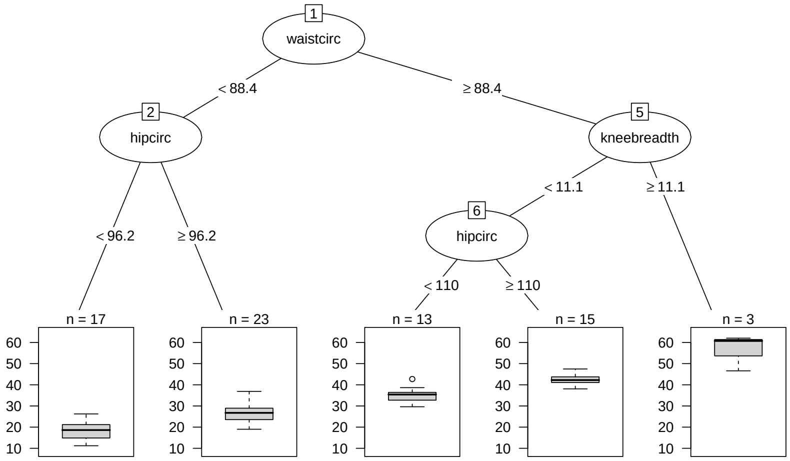

Figure 9.2 Pruned regression tree for body fat data.

```
+      control = rpart.control(xval = 100))
R> glaucoma_rpart$cptable

      CP nsplit rel error xerror   xstd
1 0.6531      0     1.000  1.531 0.0605
2 0.0714      1     0.347  0.388 0.0565
3 0.0136      2     0.276  0.378 0.0559
4 0.0100      5     0.235  0.449 0.0596

R> opt <- which.min(glaucoma_rpart$cptable[,"xerror"])
R> cp <- glaucoma_rpart$cptable[opt, "CP"]
R> glaucoma_prune <- prune(glaucoma_rpart, cp = cp)
```

As we discussed earlier, the choice of the appropriately sized tree is not a trivial problem. For the glaucoma data, the above choice of three leaves is very unstable across multiple runs of cross-validation. As an illustration of this problem we repeat the very same analysis as shown above and record the optimal number of splits as suggested by the cross-validation runs.

```
R> nsplitopt <- vector(mode = "integer", length = 25)
R> for (i in 1:length(nsplitopt)) {
+      cp <- rpart(Class ~ ., data = GlaucomaM)$cptable
+      nsplitopt[i] <- cp[which.min(cp[,"xerror"]), "nsplit"]
+  }
```

```
R> DEXfat_pred <- predict(bodyfat_prune, newdata = bodyfat)
R> xlim <- range(bodyfat$DEXfat)
R> plot(DEXfat_pred ~ DEXfat, data = bodyfat, xlab = "Observed",
+       ylab = "Predicted", ylim = xlim, xlim = xlim)
R> abline(a = 0, b = 1)
```

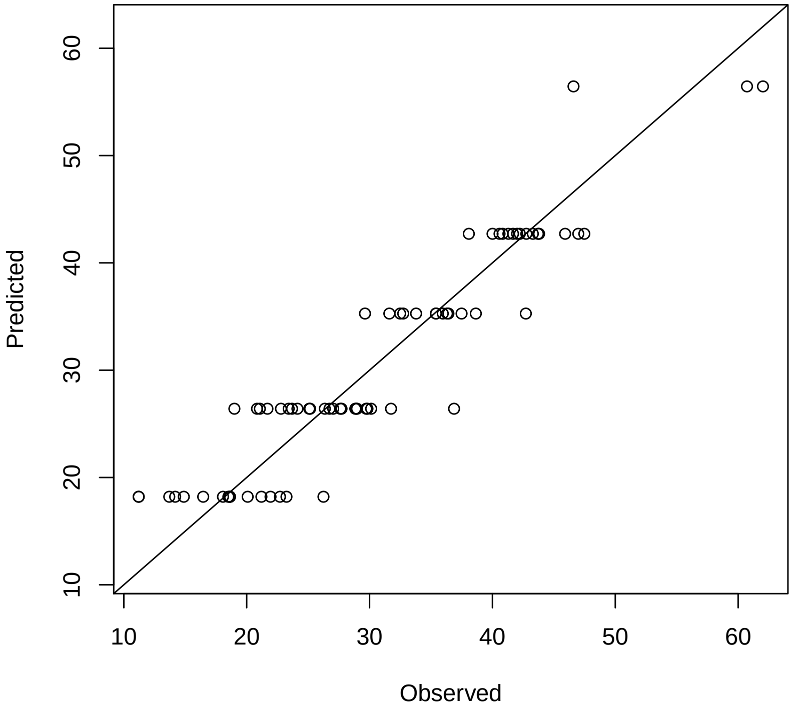

Figure 9.3 Observed and predicted DXA measurements.

```
R> table(nsplitopt)
```

```
nsplitopt
 1  2  5
14  7  4
```

Although for 14 runs of cross-validation a simple tree with one split only is suggested, larger trees would have been favored in 11 of the cases. This short analysis shows that we should not trust the tree in Figure 9.4 too much.

One way out of this dilemma is the aggregation of multiple trees via bagging. In R, the bagging idea can be implemented by three or four lines of code. Case

```
R> plot(as.party(glaucoma_prune), tp_args = list(id = FALSE))
```

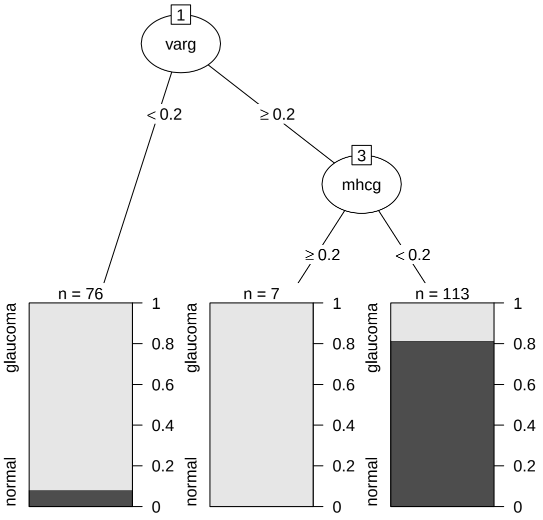

Figure 9.4 Pruned classification tree of the glaucoma data with class distribution in the leaves.

count or weight vectors representing the bootstrap samples can be drawn from the multinominal distribution with parameters n and $p_1 = 1/n, \dots, p_n = 1/n$ via the `rmultinom` function. For each weight vector, one large tree is constructed without pruning and the *rpart* objects are stored in a list, here called `trees`:

```
R> trees <- vector(mode = "list", length = 25)
R> n <- nrow(GlaucomaM)
R> bootsamples <- rmultinom(length(trees), n, rep(1, n)/n)
R> mod <- rpart(Class ~ ., data = GlaucomaM,
+              control = rpart.control(xval = 0))
R> for (i in 1:length(trees))
+      trees[[i]] <- update(mod, weights = bootsamples[,i])
```

The `update` function re-evaluates the call of `mod`, however, with the weights being altered, i.e., fits a tree to a bootstrap sample specified by the weights. It is interesting to have a look at the structures of the multiple trees. For example, the variable selected for splitting in the root of the tree is not unique as can be seen by

```
R> table(sapply(trees, function(x) as.character(x$frame$var[1])))
```

```
phcg varg vari vars
   1   14    9    1
```

Although `varg` is selected most of the time, other variables such as `vari` occur as well – a further indication that the tree in Figure 9.4 is questionable and that hard decisions are not appropriate for the glaucoma data.

In order to make use of the ensemble of trees in the list `trees` we estimate the conditional probability of suffering from glaucoma given the covariates for each observation in the original data set by

```
R> classprob <- matrix(0, nrow = n, ncol = length(trees))
R> for (i in 1:length(trees)) {
+      classprob[,i] <- predict(trees[[i]],
+                               newdata = GlaucomaM)[,1]
+      classprob[bootsamples[,i] > 0,i] <- NA
+  }
```

Thus, for each observation we get 25 estimates. However, each observation has been used for growing one of the trees with probability 0.632 and thus was not used with probability 0.368. Consequently, the estimate from a tree where an observation was not used for growing is better for judging the quality of the predictions and we label the other estimates with `NA`.

Now, we can average the estimates and we vote for glaucoma when the average of the estimates of the conditional glaucoma probability exceeds 0.5. The comparison between the observed and the predicted classes does not suffer from overfitting since the predictions are computed from those trees for which each single observation was *not* used for growing.

```
R> avg <- rowMeans(classprob, na.rm = TRUE)
R> predictions <- factor(ifelse(avg > 0.5, "glaucoma",
+                                          "normal"))
R> predtab <- table(predictions, GlaucomaM$Class)
R> predtab
```

```
predictions glaucoma normal
   glaucoma       77     12
   normal         21     86
```

Thus, an honest estimate of the probability of a glaucoma prediction when the patient is actually suffering from glaucoma is

```
R> round(predtab[1,1] / colSums(predtab)[1] * 100)
```

```
glaucoma
      79
```

per cent. For

```
R> round(predtab[2,2] / colSums(predtab)[2] * 100)
```

```
normal
    88
```

percent of normal eyes, the ensemble does not predict glaucomateous damage.

The bagging procedure is a special case of a more general approach called *random forest* (Breiman, 2001). The package **randomForest** (Breiman et al., 2013) can be used to compute such ensembles via

```
R> library("lattice")
R> gdata <- data.frame(avg = rep(avg, 2),
+      class = rep(as.numeric(GlaucomaM$Class), 2),
+      obs = c(GlaucomaM[["varg"]], GlaucomaM[["vari"]]),
+      var = factor(c(rep("varg", nrow(GlaucomaM)),
+                     rep("vari", nrow(GlaucomaM)))))
R> panelf <- function(x, y) {
+             panel.xyplot(x, y, pch = gdata$class)
+             panel.abline(h = 0.5, lty = 2)
+         }
R> print(xyplot(avg ~ obs | var, data = gdata,
+         panel = panelf,
+         scales = "free", xlab = "",
+         ylab = "Estimated Class Probability Glaucoma"))
```

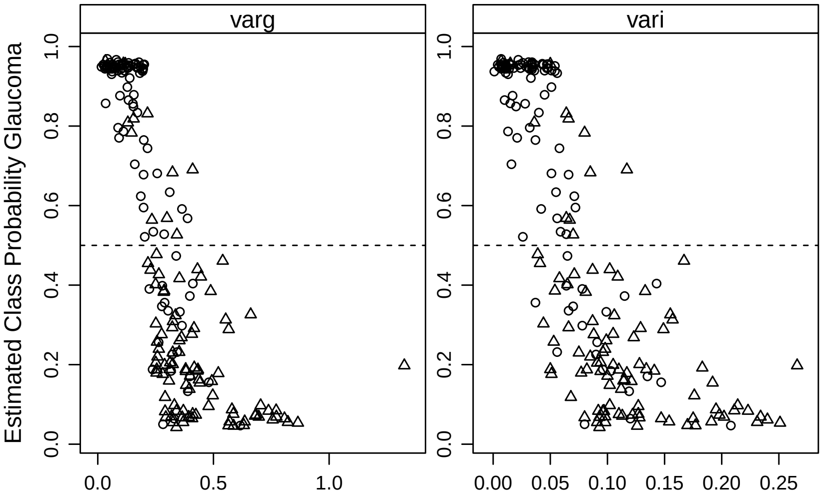

Figure 9.5 Estimated class probabilities depending on two important variables. The 0.5 cut-off for the estimated glaucoma probability is depicted as a horizontal line. Glaucomateous eyes are plotted as circles and normal eyes are triangles.

```
R> plot(bodyfat_ctree, tp_args = list(id = FALSE))
```

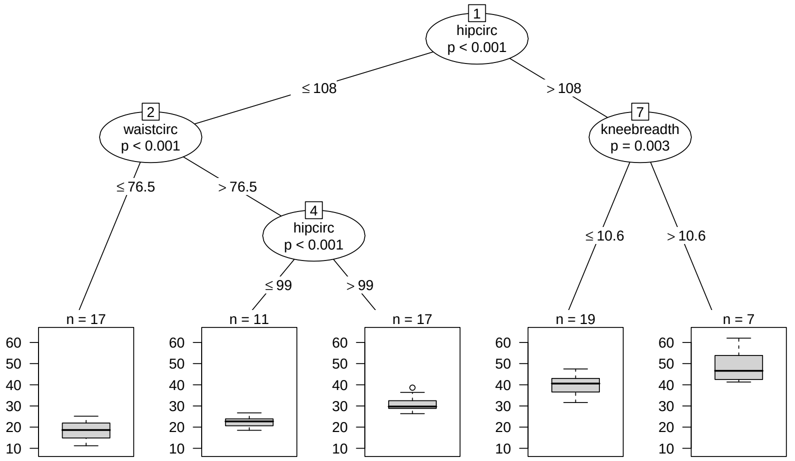

Figure 9.6 Conditional inference tree with the distribution of body fat content shown for each terminal leaf.

```
R> library("randomForest")
R> rf <- randomForest(Class ~ ., data = GlaucomaM)
```

and we obtain out-of-bag estimates for the prediction error via

```
R> table(predict(rf), GlaucomaM$Class)

           glaucoma normal
  glaucoma       80     11
  normal         18     87
```

9.3.3 Trees Revisited

For the body fat data, such a *conditional inference tree* can be computed using the `ctree` function

```
R> bodyfat_ctree <- ctree(DEXfat ~ age + waistcirc + hipcirc +
+      elbowbreadth + kneebreadth, data = bodyfat)
```

This tree doesn't require a pruning procedure because an internal stop criterion based on formal statistical tests prevents the procedure from overfitting the data. The tree structure is shown in Figure 9.6. Although the structure of this tree and the tree depicted in Figure 9.2 are rather different, the corresponding predictions don't vary too much.

Very much the same code is needed to grow a tree on the glaucoma data:

```
R> glaucoma_ctree <- ctree(Class ~ ., data = GlaucomaM)
```

and a graphical representation is depicted in Figure 9.7 showing both the cutpoints and the *p*-values of the associated independence tests for each node.

```
R> plot(glaucoma_ctree, tp_args = list(id = FALSE))
```

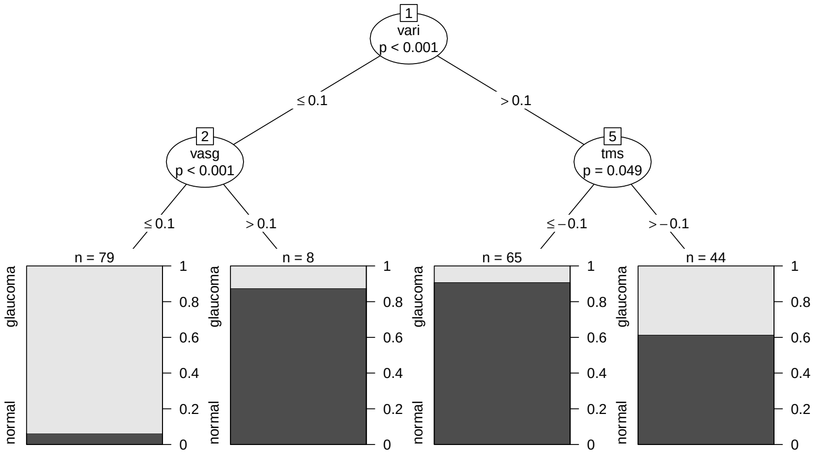

Figure 9.7 Conditional inference tree with the distribution of glaucomateous eyes shown for each terminal leaf.

The first split is performed using a cutpoint defined with respect to the volume of the optic nerve above some reference plane, but in the inferior part of the eye only (`vari`).

9.3.4 Happiness in China

A conditional inference tree is a simple alternative to the proportional odds model for the regression analysis of the happiness variable from the Chinese Health and Family Life Survey. In each node, a linear association test introduced in Section **??** taking the ordering of the happiness levels into account is applied for selecting variables and split-points. Before we fit the tree with the `ctree` function, we recode the levels of the happiness variable to allow plotting of these symbols with restricted page space:

```
R> plot(CHFLS_ctree, ep_args = list(justmin = 10),
+       tp_args = list(id = FALSE))
```

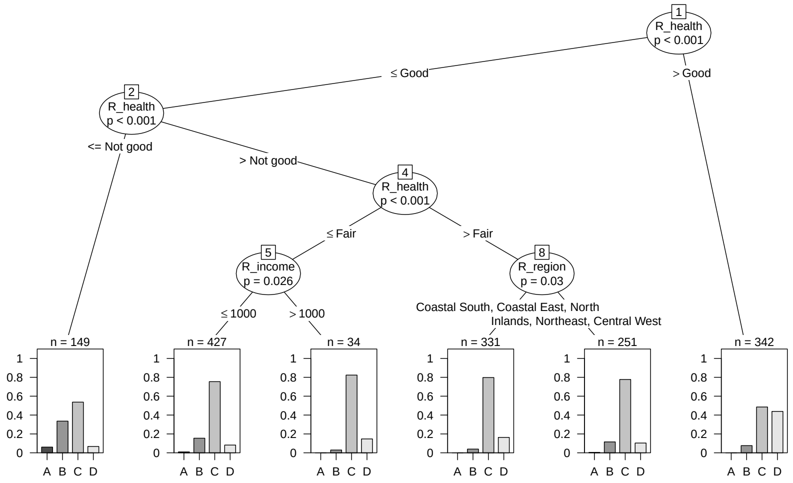

Figure 9.8 Conditional inference tree with the distribution of self-reported happiness shown for each terminal leaf. The levels of happiness have been abbreviated (A: very unhappy, B: not too happy, C: somewhat happy; D: very happy). The `justmin` argument ensures that split descriptions longer than 10 characters are displayed over two lines.

```
R> levels(CHFLS$R_happy)

[1] "Very unhappy"   "Not too happy"  "Somewhat happy"
[4] "Very happy"

R> levels(CHFLS$R_happy) <- LETTERS[1:4]
R> CHFLS_ctree <- ctree(R_happy ~ ., data = CHFLS)
```

The resulting tree is depicted in Figure 9.8 and very nicely backs up the results obtained from the proportional odds model in Chapter **??**. The distribution of self-reported happiness is shifted from very unhappy to very happy with increasing values of self-reported health, i.e., women that reported excellent health (mind the > sign in the right label of the root split!) were at least somewhat happy with only a few exceptions. Women with poor or not good health reported being not too happy much more often. There seems to be further differentiation with respect to geography and also income but the differences in the distributions depicted in the terminal leaves are negligible.

Bibliography

Breiman, L. (2001), "Random forests," *Machine Learning*, 45, 5–32.

Breiman, L., Cutler, A., Liaw, A., and Wiener, M. (2013), ***randomForest**: Breiman and Cutler's Random Forests for Classification and Regression*, URL `http://stat-www.berkeley.edu/users/breiman/RandomForests`, R package version 4.6-7.

Hothorn, T. and Zeileis, A. (2014), ***partykit**: A Toolkit for Recursive Partytioning*, URL `http://R-forge.R-project.org/projects/partykit/`, R package version 0.8-0.