Data Analysis & Graphics Using R – Solutions to Exercises (April 24, 2004)

*Preliminaries*

```
> library(DAAG)
> library(rpart)
```

*Exercise 1*
Refer to the `head.injury` data frame.

(a) Use the default setting in `rpart()` to obtain a tree-based model for predicting occurrence of clinically important brain injury, given the other variables.

(b) How many splits gives the minimum cross-validation error?

(c) Prune the tree using the 1 standard error rule.

(a)
```
> data(head.injury)
> set.seed(29)
> injury.rpart <- rpart(clinically.important.brain.injury ~ .,
+     data = head.injury, method = "class", cp = 1e-04)
> plotcp(injury.rpart)
> printcp(injury.rpart)

Classification tree:
rpart(formula = clinically.important.brain.injury ~ ., data = head.injury,
    method = "class", cp = 1e-04)

Variables actually used in tree construction:
[1] GCS.13               GCS.15.2hours        age.65
[4] amnesia.before       basal.skull.fracture high.risk
[7] loss.of.consciousness vomiting

Root node error: 250/3121 = 0.0801

n= 3121

      CP nsplit rel error xerror   xstd
1 0.0400      0     1.000  1.000 0.0607
2 0.0360      2     0.920  0.992 0.0604
3 0.0140      3     0.884  0.896 0.0577
4 0.0080      5     0.856  0.904 0.0579
5 0.0001     10     0.816  0.916 0.0583
```

The setting `cp=0.0001` was reached after some experimentation.

(b) The minimum cross-validated relative error is for `nsplit=3`, i.e., for a tree size of 4.

(c) The one-standard-error rule likewise chooses `nsplit=3`, with `cp=0.014`. Setting `cp=0.02`, i.e., larger than `cp` for the next smallest number of splits, will prune the tree back to this size. We have

```
> injury0.rpart <- prune(injury.rpart, cp = 0.02)
```

We plot the tree from (a) that shows the cross-validated relative error, and the tree obtained from (c).

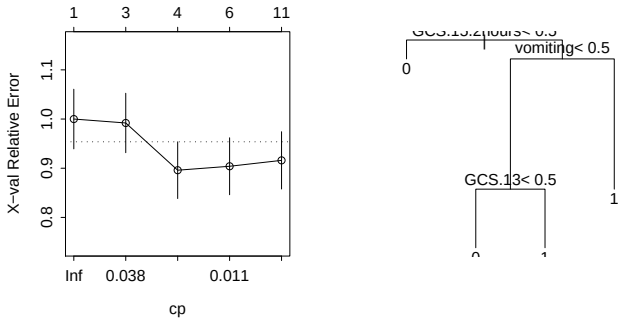


Figure 1: Plots: (i) of cross-validated relative error versus `cp`; and (ii) of the tree obtained in (c). Both plots are for the head injury data.

There can be substantial change from one run to the next.

*Exercise 2*
The data set `mifem` is part of the larger data set in the data frame `monica` that we have included in our *DAAG* package. Use tree-based regression to predict mortality in this larger data set. What is the most immediately striking feature of your analysis? Should this be a surprise?

```
> data(monica)
> monica.rpart <- rpart(outcome ~ ., data = monica, method = "class")
```

```
> plot(monica.rpart)
> text(monica.rpart)
```

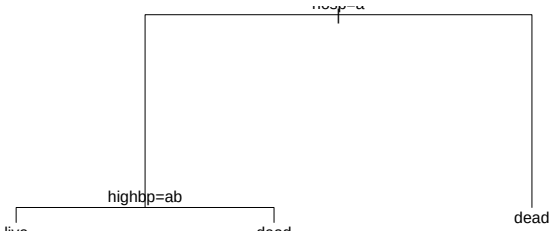


Figure 2: Classification tree for monica data.

Those who were not hospitalised were very likely to be dead! Check by examining the table:

```
> table(monica$hosp, monica$outcome)
```

```
    live dead
  y 3522  920
  n    3 1922
```

*Exercise 3*
Use tree-based regression to predict `re78` in the data frame `nsw74pred1` that is in our *DAAG* package. Compare the predictions with the multiple regression predictions in Chapter 6.

In order to reproduce the same results as given here, do:

```
> set.seed(21)
```

Code for the initial calculation is:

```
> data(nsw74psid1)
> nsw.rpart <- rpart(re78 ~ ., data = nsw74psid1, cp = 0.001)
> plotcp(nsw.rpart)
```

It is obvious that `cp=0.002` will be adequate. At this point, the following is a matter of convenience, to reduce the printed output:

```
> nsw.rpart <- prune(nsw.rpart, cp = 0.002)
> printcp(nsw.rpart)

Regression tree:
rpart(formula = re78 ~ ., data = nsw74psid1, cp = 0.001)

Variables actually used in tree construction:
[1] age  educ re74 re75

Root node error: 6.53e+11/2675 = 2.44e+08

n= 2675

        CP nsplit rel error xerror   xstd
1  0.34463      0     1.000  1.001 0.0463
2  0.11009      1     0.655  0.665 0.0390
3  0.04094      2     0.545  0.558 0.0330
4  0.03178      3     0.504  0.518 0.0352
5  0.01582      4     0.473  0.506 0.0346
6  0.01057      5     0.457  0.491 0.0347
7  0.01053      6     0.446  0.485 0.0345
8  0.00633      7     0.436  0.469 0.0325
9  0.00566      8     0.429  0.460 0.0330
10 0.00388      9     0.424  0.461 0.0331
11 0.00355     10     0.420  0.462 0.0331
12 0.00318     11     0.416  0.473 0.0338
13 0.00283     12     0.413  0.475 0.0337
14 0.00272     13     0.410  0.475 0.0338
15 0.00233     15     0.405  0.476 0.0338
16 0.00202     16     0.402  0.476 0.0336
17 0.00200     17     0.400  0.477 0.0339
```

The minimum cross-validated relative error is at `nsplit=12`. The one standard error limit is 0.498 (=0.463+0.035). The one standard error rule suggests taking `nsplit=5`.

If we go with the one standard error rule, we have a residual variance equal to $244284318 \times 0.49177 = 120131699$.

For the estimate of residual variance from the calculations of Section 6.x, we do the following.

```
> data(nsw74psid1)
> attach(nsw74psid1)
> here <- age <= 40 & re74 <= 5000 & re75 <= 5000 & re78 < 30000
> nsw74psidA <- nsw74psid1[here, ]
> detach(nsw74psid1)
> A1.lm <- lm(re78 ~ trt + (age + educ + re74 + re75) + (black +
+     hisp + marr + nodeg), data = nsw74psidA)
> summary(A1.lm)$sigma^2

[1] 40177577
```

The variance estimate is 40177577. This is about a third of the variance estimate that was obtained with tree-based regression.

*Exercise 4*
Copy down the email spam data set from the web site given in Section 10.2. Carry out a tree-based regression using all 57 available explanatory variables. Determine the change in the cross-validation estimate of predictive accuracy.

We set the random number seed to 21, to allow users to reproduce our results. In most other contexts, it will be best not to set a seed. The file **spam.shortnames** is available for copying from the web address `http://wwwmaths.anu.edu.au/ johnm/r-book/xtra-data`. The data frame `spam` is created thus:

```
> spam <- read.table("spambase.data", header = FALSE, sep = ",")
> nam <- scan("spam.shortnames", what = "")
> names(spam) <- nam
```

Now load *rpart* and proceed with the calculations.

```
> set.seed(21)
> spam.rpart <- rpart(yesno ~ ., data = spam, cp = 1e-04, method = "class")
> printcp(spam.rpart)

Classification tree:
rpart(formula = yesno ~ ., data = spam, method = "class", cp = 1e-04)

Variables actually used in tree construction:
 [1] '1999'     '650'      address    bang       crl.av     crl.long
 [7] crl.tot    data       dollar     edu        email      font
[13] free       george     hp         internet   leftparen  money
[19] our        over       re         remove     semicolon  technology
[25] will       you        your

Root node error: 1813/4601 = 0.394

n= 4601

         CP nsplit rel error xerror    xstd
1  0.476558      0     1.000  1.000 0.01828
2  0.148924      1     0.523  0.558 0.01550
3  0.043023      2     0.375  0.460 0.01441
4  0.030888      4     0.288  0.322 0.01245
```

```
5  0.010480      5     0.258  0.279 0.01171
6  0.008274      6     0.247  0.266 0.01146
7  0.007170      7     0.239  0.261 0.01136
8  0.005295      8     0.232  0.253 0.01121
9  0.004413     14     0.196  0.236 0.01086
10 0.003585     15     0.191  0.228 0.01071
11 0.002758     19     0.177  0.226 0.01065
12 0.002574     22     0.169  0.222 0.01058
13 0.002206     25     0.161  0.222 0.01058
14 0.002114     27     0.157  0.219 0.01051
15 0.001655     33     0.144  0.212 0.01035
16 0.001103     36     0.139  0.199 0.01005
17 0.000827     43     0.131  0.196 0.00998
18 0.000552     47     0.128  0.196 0.00998
19 0.000368     53     0.125  0.201 0.01010
20 0.000100     62     0.121  0.201 0.01010
```

Figure 3 shows the graph that is obtained by plotting this tree. For making a decision on the size of tree however, it is convenient to work from the information given by the function `printcp()`.

```
> plotcp(spam.rpart)
```

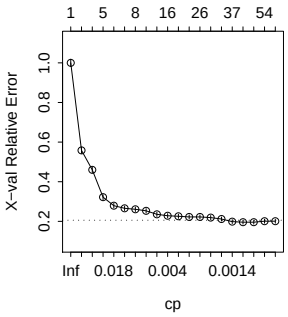


Figure 3: Plot of cross-validated relative error versus `cp`, for the full `spam` data set.

Setting `cp=0.0001` ensures, when the random number seed is set to 21, that the cross-validated relative error reaches a minimum, of 0.1958, at `nsplit=43`. Pruning to get the tree that is likely to have best predictive power can use `cp=0.001`. Adding the SE to the minimum cross-validated relative error gives 0.2. The smallest tree with an SE smaller than this is at `nsplit=36`; setting `cp=0.0012` will give this tree.

Here then are the two prunings:

```
> spam.rpart1 <- prune(spam.rpart, cp = 0.001)
> spam.rpart2 <- prune(spam.rpart, cp = 0.0012)
```