Regression Trees

Aim of tutorial:

- Use rpart() to build regression trees
  - Example 1 in lecture notes is considered (chapter 6).
  - Use the 1-SE rule to prune trees

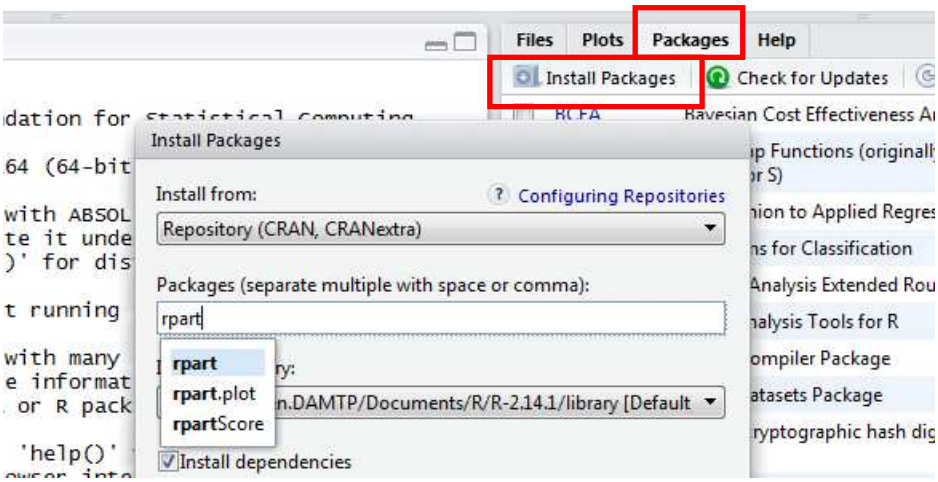
Regression Trees

There are a number of packages within R capable of producing decision trees (classification and/or regression trees). Packages include tree() and rpart(). This tutorial will focus on using rpart(), which largely follows the CART algorithm.

i. Installing/Loading the package

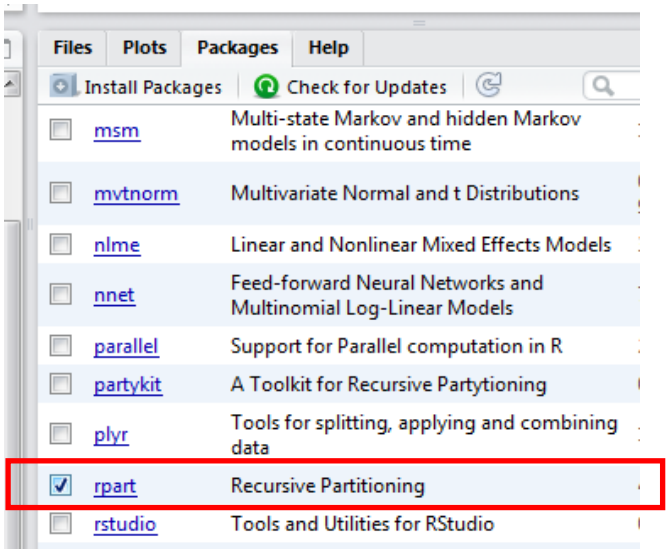
In order to be able to produce regression trees the package rpart() needs to be installed on your machine.

Look at the list of Packages available under the Packages tab (within RStudio), and if rpart() is not on the list, select Install Packages and type in 'rpart'



The package also needs to be loaded into a working session, either through ticking the Package name or by the following command:

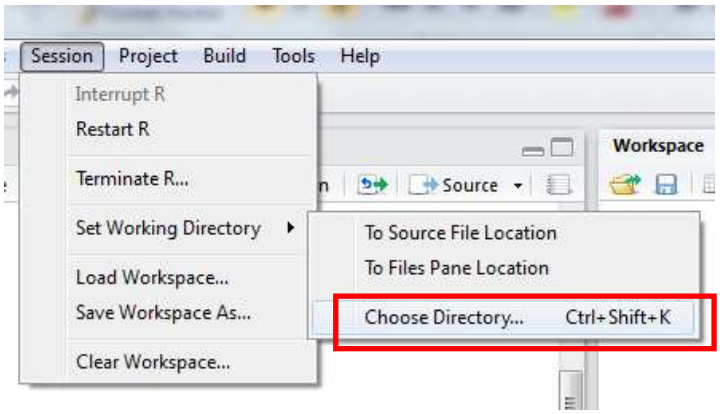
```
> library("rpart")
```



ii. Accessing Example 1 data

The example data is stored in a csv file called worked_example_CART.csv. This should be downloaded from QOL and saved within your Q drive.

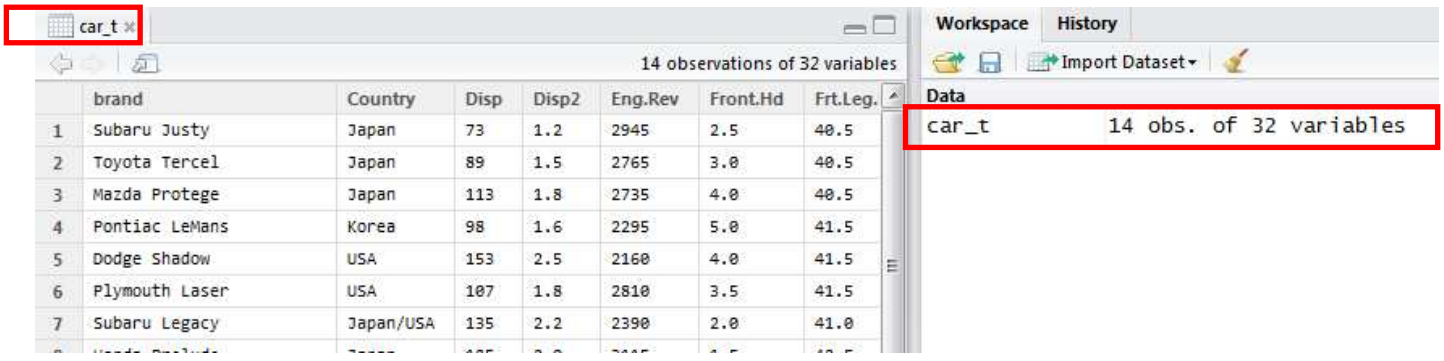
To be able to access this data, set the working directory to the folder where you have saved the csv file.



Submit the follow code to load the data:

```
> car_t<-read.csv("worked_example_CART.csv")
```

Look at the data by selecting it from the Workspace window and viewing its contents:



Notice this contains the 14 observations given in Example 1 of Chapter 6. [It also contains additional variables present in the full data set, which were not mentioned in Example 1.]

In order to build the regression tree, we select the variables that are to be considered as input. We wish to ensure that the car brands are not used in the regression tree (this is an ID variable), so it can be removed using the following R code:

```
> cars_t2 <- car_t[, -match(c("brand"), names(car_t))]
```

[The minus sign in front of match indicates that the "brand" variable is to be removed.]

iii. Building a Regression Tree

Consider submitting the following code to model the price (in $1000) of the cars:

car_rt <- rpart(Price/1000 ~ ., data=cars_t2,method = "anova", x=TRUE, y=TRUE, control = rpart.control(minsplit=4,xval=10,cp = 0))

- Note that the dependent variable is Price/1000.
- The use of the "." after the "~" indicates that all variables within the data set are to be considered. This can be replaced by a specified list.
- The "data=" statement indicates what data set is being condering.
- method="anova" indicates that a regression tree is to be built (as opposed to a classification tree).
- X=TRUE, Y=TRUE, indicate the inputs and target variable are kept in the output of rpart.
- The 'control' element is modifying some of the default rpart settings that control aspects of the rpart fit.
- By default, the minimum number of observations that must exist in a node in order for a split to be attempted (minsplit) equals 20. However since this sample data has only 14 observations, to ensure a tree is built, minsplit was set to a much smaller number.
- xval controls the *k* of the *k*-fold cross validation.
- The complexity parameter (cp) is used to decide how far the tree needs pruned back. Setting cp=0, ensures that the tree is built in its entirety (pruning does not occur) – only other settings should as minsplit will result in leaves being produced in the regression tree.

To see some of the output:

```
> car_rt
n= 14

node), split, n, deviance, yval
      * denotes terminal node

 1) root 14 411.8976000 13.182930
   2) Weight< 2675 5   5.1769010  7.010400
     4) Eng.Rev>=2227.5 4   0.9697047  6.551750
       8) Country=Japan 3   0.3121647  6.317667 *
       9) Country=Korea 1   0.0000000  7.254000 *
     5) Eng.Rev< 2227.5 1   0.0000000  8.845000 *
   3) Weight>=2675 9 110.3865000 16.612110
     6) HP< 145 4  35.5603000 13.687250
      12) Tank>=15.85 3   5.3152510 12.099670 *
      13) Tank< 15.85 1   0.0000000 18.450000 *
     7) HP>=145 5  13.2315600 18.952000
      14) HP< 162.5 3   0.2673260 17.679000 *
      15) HP>=162.5 2   0.8102645 20.861500 *
```

- Notice that at the root node, there are 14 observations, the deviance=411.9 agreeing with the deviance of 'nosplits' calculated in example 1 (p3). The yval indicates that if the tree was stopped here (i.e. has no splits) the fit would suggest the price in \$1000 equals 13.18 for all cars.
- Below the root node the optimal split is found to be based on Weight<2675/Weight≥2675. Again if the tree was stopped at this point, the prediction on the Weight<2675 branch would be 7.01, while that along the other branch would be 16.6 (see p4 of chapter 6). The deviance of each subset is given here e.g.

$$D_{\text{group1}} = \sum_{i=1}^{5} \left(y_i - \bar{y}_{\text{group1}} \right)^2$$
$$= \left(5.866 - 7.01\right)^2 + \ldots + \left(8.845 - 7.01\right)^2 = 5.177$$
$$D_{\text{group2}} = \sum_{i=6}^{14} \left(y_i - \bar{y}_{\text{group2}} \right)^2$$
$$= \left(17.9 - 16.61\right)^2 + \ldots + \left(17.257 - 16.61\right)^2 = 110.387$$
$$D_{\text{with split}} = 5.18 + 110.39 = 115.56$$

  (in agreement with $D_{\text{with split}}$ calculated in example 1 (p5 of chapter 6).
- The information above also list further splits within each sub-group e.g. in the weight<2675 subgroup data is next split by Eng.Rev>=2227.5/.Rev<2227.5.

Further output can be seen using:

```
> summary(car_rt)
Call:
rpart(formula = Price/1000 ~ ., data = cars_t2, method = "anova",
    x = TRUE, y = TRUE, control = rpart.control(minsplit = 4,
        xval = 10, cp = 0))
  n= 14

          CP nsplit  rel error    xerror      xstd
1 0.719436548      0 1.00000000 1.161501 0.1973853
2 0.149538748      1 0.28056345 1.302108 0.3902376
3 0.073428568      2 0.13102470 1.378413 0.3468871
4 0.029507256      3 0.05759614 1.633062 0.4500611
5 0.010214181      4 0.02808888 1.342072 0.3955020
6 0.001596368      5 0.01787470 1.375601 0.4067138
7 0.000000000      6 0.01627833 1.388632 0.4137170

Variable importance
       Tank       Weight           HP Frt.Leg.Room
         21           20           17           12
      Width         Disp        Disp2     Frt.Shld
         12           10            2            2
    Country
          2

Node number 1: 14 observations,    complexity param=0.7194365
  mean=13.18293, MSE=29.42126
  left son=2 (5 obs) right son=3 (9 obs)
  Primary splits:
      Weight     < 2675   to the left,  improve=0.719436, (0 mis
      HP         < 108.5  to the left,  improve=0.6287782, (0 mis
      Type       splits as  RRRLR,      improve=0.5978335, (0 mis
      Length     < 184    to the left,  improve=0.5183372, (0 mis
      Wheel.base < 103    to the left,  improve=0.5183372, (0 mis
  Surrogate splits:
      Tank         < 14.25  to the left,  agree=0.929, adj=0.8, (
split)
      Frt.Leg.Room < 40.75  to the left,  agree=0.857, adj=0.6, (
split)
      HP           < 85     to the left,  agree=0.857, adj=0.6, (
split)
      Width        < 66.5   to the left,  agree=0.857, adj=0.6, (
split)
      Disp         < 102.5  to the left,  agree=0.786, adj=0.4, (
split)

Node number 2: 5 observations,    complexity param=0.01021418
  mean=7.0104, MSE=1.03538
  left son=4 (4 obs) right son=5 (1 obs)
  Primary splits:
      Eng.Rev    < 2227.5 to the right, improve=0.8126863, (0 mis
      Gear.Ratio < 3.575  to the right, improve=0.8126863, (0 mis
```

This table gives information useful for deciding the complexity parameter (see later)

Notice improve agrees with that calculated (p5)

Agrees with that calculated (p5)

Notice that other possible splits are listed – in rank order to how well they improve the model

Notice that the Complexity Parameter is given for each node (node 1 agrees with that calculated (p5)). The complexity parameter at the node 2 was calculated using:

$$\alpha_{crit} = \frac{D_{\text{without split}} - D_{\text{with split}}}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = \frac{0.8126863 \times 5.1769}{411.89759} = 0.0102$$

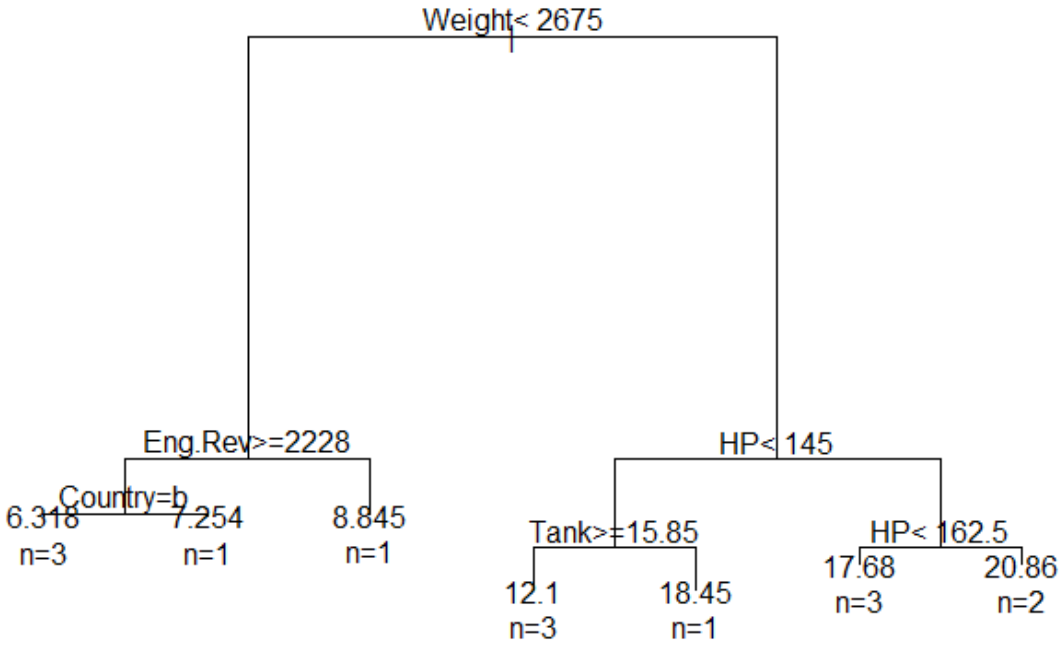
Notice that surrogate variables are also listed:

When input variables required by the tree (split variables) are missing in data, surrogate variables allow these missing inputs to be estimated using the other independent variables. As an example, assume that the split (weight<2675,weight ≥ 2675) has been chosen. The surrogate variables are found by re-applying the partitioning algorithm (without recursion) to predict the two categories weight<2675' vs. ' weight ≥ 2675 ' using the other independent variables. The surrogates are ranked according to how well each correctly predicts the (weight<2675,weight ≥ 2675) split. Any observation which is missing the split variable is then classified using the first surrogate variable, or if missing that, the second surrogate is used etc.

To visually view the tree submit:

```
plot(car_rt, compress = FALSE, margin=0.05)
text(car_rt, use.n = TRUE)
```

Producing:



iv. Pruning a Regression Tree

Notice that the CP table indicates that if CP is above 0.1495, then the numbers of splits will equal 1:

```
           CP nsplit  rel error   xerror      xstd
1 0.719436548      0 1.00000000 1.161501 0.1973853
2 0.149538748      1 0.28056345 1.302108 0.3902376
3 0.073428568      2 0.13102470 1.378413 0.3468871
4 0.029507256      3 0.05759614 1.633062 0.4500611
5 0.010214181      4 0.02808888 1.342072 0.3955020
6 0.001596368      5 0.01787470 1.375601 0.4067138
7 0.000000000      6 0.01627833 1.388632 0.4137170
```

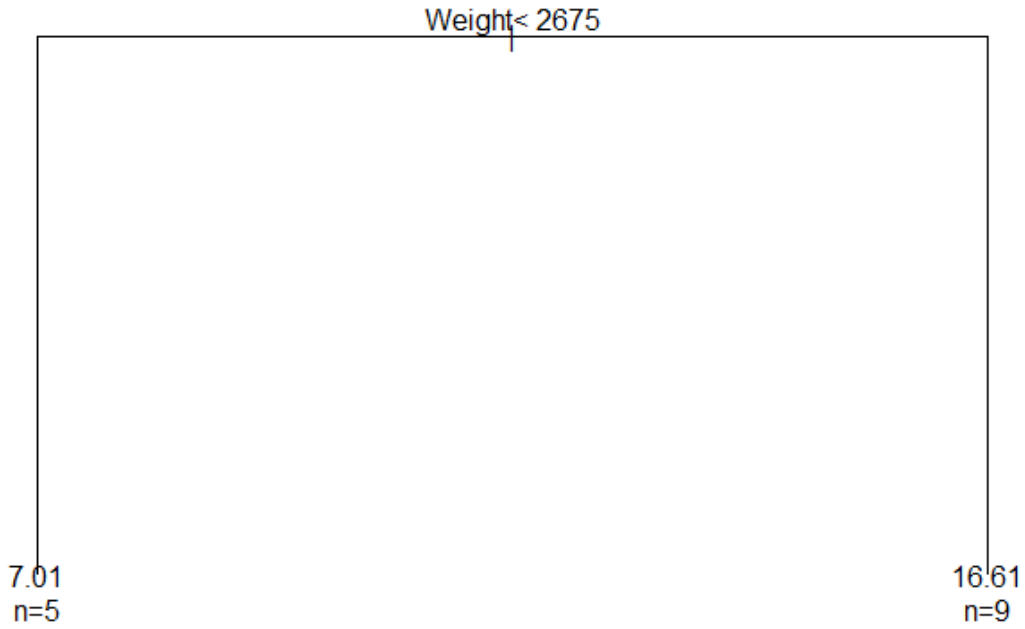
Thus submit the following code to get this pruned tree:

```
> prune_rt <- prune(car_rt, cp = 0.26)
> prune_rt
n= 14

node), split, n, deviance, yval
      * denotes terminal node

1) root 14 411.897600 13.18293
  2) Weight< 2675 5   5.176901  7.01040 *
  3) Weight>=2675 9 110.386500 16.61211 *
> plot(prune_rt, compress = FALSE, margin=0.05)
> text(prune_rt, use.n = TRUE)
```

Producing:



v. Building a Regression Tree Using the Entire Data Set: 1-SE Rule

These 14 observations were a sub-sample of the car90 dataset. Load these into R using:

```
> data(car90, package = "rpart")
> cars <- car90[, -match(c("Rim", "Tires", "Model2"), names(car90))]
```

Next submit the following code to build a regression tree on this entire data set:

```
set.seed(1)
#this ensure the errors calculated from the k-fold cross validation are consistent with
#those shown below in this tutorial
carfit <- rpart(Price/1000 ~ ., data=cars,method = "anova", x=TRUE, y=TRUE, control
= rpart.control(xval=111, cp = 0))
```

Note that xval=111 corresponds to 111-fold cross validation or **leave-one-out** (since there are 111 observations in the data).

To decide on the appropriate value of the complexity parameter, the 1-SE rule can be employed. This rule considers the error found via the *k*-fold cross validation as the complexity parameter is varied. The information used in the rule is found in the cp table:

```
> printcp(carfit)

Regression tree:
rpart(formula = Price/1000 ~ ., data = cars, method = "anova",
    x = TRUE, y = TRUE, control = rpart.control(xval = 111, cp =
0))

Variables actually used in tree construction:
[1] Country Disp    HP.revs Luggage Tank    Type

Root node error: 7118.3/105 = 67.793

n=105 (6 observations deleted due to missingness)

         CP nsplit rel error  xerror    xstd
1 0.4601461      0   1.00000 1.01932 0.16312
2 0.1179053      1   0.53985 0.81179 0.11966
3 0.1123434      2   0.42195 0.88655 0.12195
4 0.0444913      3   0.30961 0.54594 0.10909
5 0.0334494      4   0.26511 0.58898 0.12319
6 0.0047464      5   0.23166 0.55993 0.13413
7 0.0034267      6   0.22692 0.57066 0.13379
8 0.0030566      7   0.22349 0.57158 0.13373
9 0.0000000      8   0.22043 0.56500 0.13396
```

The relative error gives the total deviance of the regression tree as the number of splits is increases, with values normalised by dividing by SS_{TOT}. [So the relative error equals 1 when there are no splits.] xerror indicates the relative error that is found in the validation data as the splits are varied. Since each of the 111 validation data sets in 111-fold cross validation may give a different relative error, the spread of these relative errors in each case is indicated by the standard error (xstd).

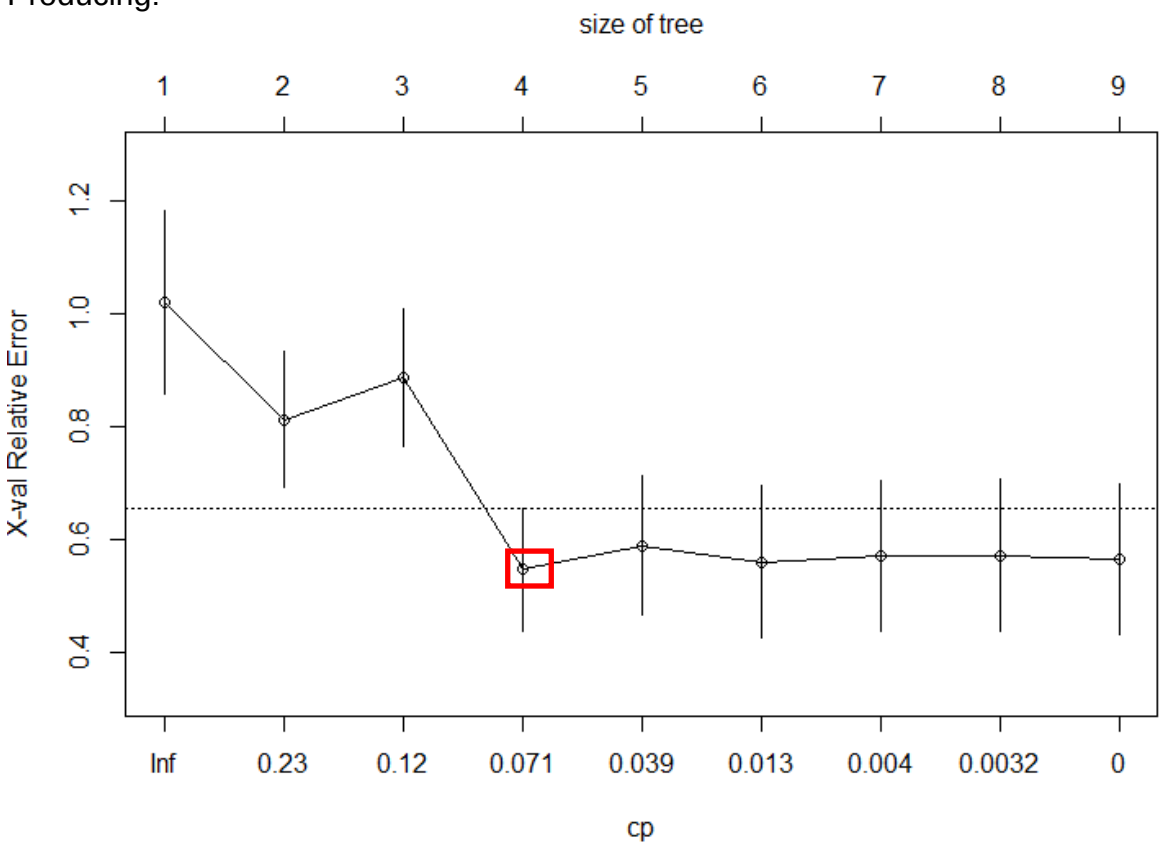
A plot of relative error in the validation data (xerror) versus complexity parameter in general often has an initial sharp drop followed by a relatively flat plateau and then a slow rise. The choice of complexity parameter among those models on the plateau can be essentially random. Using the 1-SE rule, any relative error within one standard error of the achieved minimum is marked as being equivalent to the minimum (i.e. considered to be part of the flat plateau). Then the simplest model, among all those along the plateau, is chosen. That is, a good choice of complexity parameter for pruning is often the leftmost value for which the mean xerror lies below the value of $\text{xerror}_{\min} + \text{xstd}_{\min}$.

Here $\text{xerror}_{\min} = 0.54594, \text{xstd}_{\min} = 0.10909 \Rightarrow \text{xerror}_{\min} + \text{xstd}_{\min} = 0.65503$ and so the pruned tree requires 3 splits (corresponding to 4 leaves, so the **size of the tree=4**). [With 2 splits, the xerror is too large, 0.88655>0.65503].

This can also be shown visually using:

```
plotcp(carfit, minline = TRUE, lty = 3, col = 1)
```

Producing:



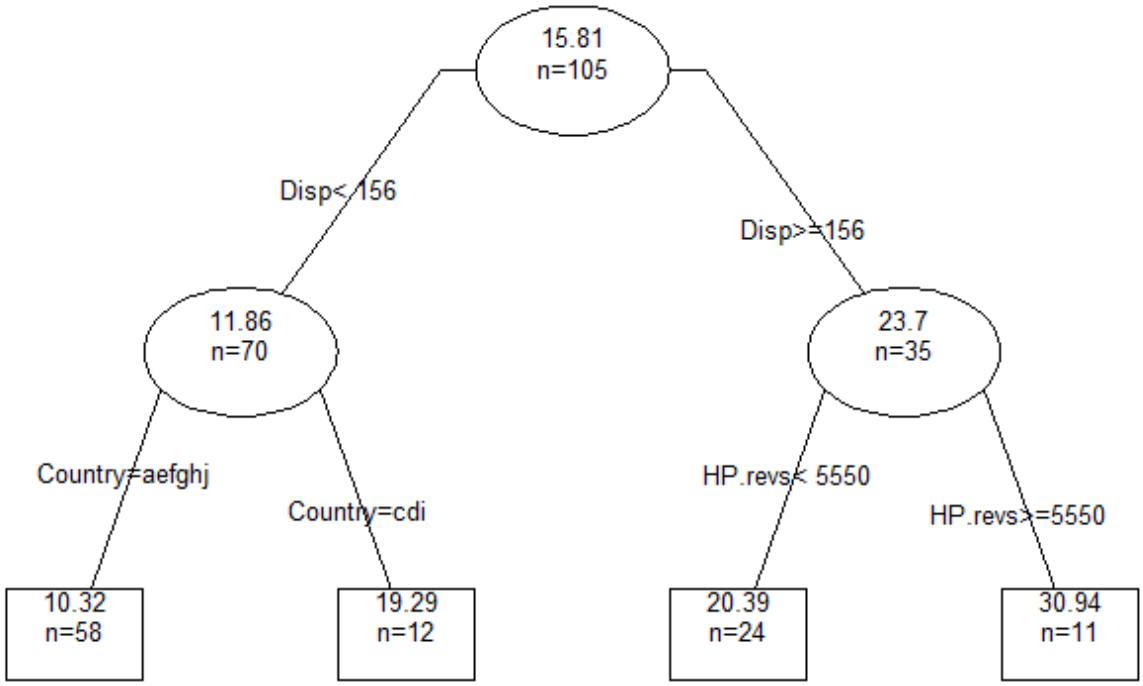
To prune the tree, set the complexity parameter to any value above 0.0444913 and below 0.1123434 and then plot the regression tree.

```
> prune_rt2 <- prune(carfit, cp = 0.05)
> prune_rt2
n=105 (6 observations deleted due to missingness)

node), split, n, deviance, yval
      * denotes terminal node

1) root 105 7118.2670 15.80522
  2) Disp< 156 70 1491.6280 11.85587
    4) Country=Brazil,Japan,Japan/USA,Korea,Mexico,USA 58  421.2147 10.31847 *
    5) Country=France,Germany,Sweden 12  270.7233 19.28667 *
  3) Disp>=156 35 2351.1960 23.70391
    6) HP.revs< 5550 24  980.2779 20.38871 *
    7) HP.revs>=5550 11  531.6368 30.93709 *
> plot(prune_rt2, uniform = TRUE, branch = 0.4, compress = TRUE, margin = 0.08)
> text(prune_rt2, all = TRUE, use.n = TRUE, fancy = TRUE, cex= 0.8)
```

Note the above plot and text statements use some additional options, to make the figure visually more appealing [use the help pages on rpart, to work out what each of the components does].


15.81
n=105
Disp< 156
Disp>=156
11.86
n=70
23.7
n=35
Country=aefghj
Country=cdi
HP.revs< 5550
HP.revs>=5550
10.32
n=58
19.29
n=12
20.39
n=24
30.94
n=11