**Tree-Based Models**

Recursive partitioning is a fundamental tool in data mining. It helps us explore the stucture of a set of data, while developing easy to visualize decision rules for predicting a categorical (classification tree) or continuous (regression tree) outcome. This section briefly describes CART modeling, conditional inference trees, and random forests.

**CART Modeling via rpart**

Classification and regression trees (as described by Brieman, Freidman, Olshen, and Stone) can be generated through the **[rpart](http://cran.r-project.org/web/packages/rpart/index.html)** package. Detailed information on **rpart**is available in [An Introduction to Recursive Partitioning Using the RPART Routines](http://www.mayo.edu/hsr/techrpt/61.pdf). The general steps are provided below followed by two examples.

**1. Grow the Tree**

To grow a tree, use  
**rpart(***formula*, **data=**, **method=,control=)** where

|  |  |
| --- | --- |
| ***formula*** | is in the format  *outcome* ~ *predictor1*+*predictor2*+*predictor3*+ect. |
| **data=** | specifies the data frame |
| **method=** | **"class"** for a classification tree  **"anova"** for a regression tree |
| **control=** | optional parameters for controlling tree growth. For example, control=rpart.control(minsplit=30, cp=0.001) requires that the minimum number of observations in a node be 30 before attempting a split and that a split must decrease the overall lack of fit by a factor of 0.001 (cost complexity factor) before being attempted. |

**2. Examine the results**

The following functions help us to examine the results.

|  |  |
| --- | --- |
| **printcp(***fit***)** | display cp table |
| **plotcp(***fit***)** | plot cross-validation results |
| **rsq.rpart(***fit***)** | plot approximate R-squared and relative error for different splits (2 plots). labels are only appropriate for the "anova" method. |
| **print(***fit***)** | print results |
| **summary(***fit***)** | detailed results including surrogate splits |
| **plot(***fit***)** | plot decision tree |
| **text(***fit***)** | label the decision tree plot |
| **post(***fit*,**file=)** | create postscript plot of decision tree |

In trees created by **rpart( )**, move to the **LEFT** branch when the stated condition is true (see the graphs below).

**3. prune tree**

Prune back the tree to avoid overfitting the data. Typically, you will want to select a tree size that minimizes the cross-validated error, the **xerror**column printed by **printcp( )**.

Prune the tree to the desired size using  
**prune(***fit*, **cp=** **)**

Specifically, use **printcp( )** to examine the cross-validated error results, select the complexity parameter associated with minimum error, and place it into the **prune( )** function. Alternatively, you can use the code fragment

**fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"]**

to automatically select the complexity parameter associated with the smallest cross-validated error. Thanks to [HSAUR](http://www.statmethods.net/about/books.html) for this idea.

**Classification Tree example 分類樹**

Let's use the data frame **kyphosis** to predict a type of deformation (kyphosis) after surgery, from age in months (Age), number of vertebrae involved (Number), and the highest vertebrae operated on (Start).

# Classification Tree with rpart  
library(rpart)

Data on Children who have had Corrective Spinal Surgery(矯正脊柱手術)

**Description**

The kyphosis (脊柱後凸) data frame has 81 rows and 4 columns. representing data on children who have had corrective spinal surgery

Kyphosis

a factor with levels absent present indicating if a kyphosis (a type of deformation) was present after the operation.

Age

in months

Number

the number of vertebrae involved涉及椎骨的數

Start

the number of the first (topmost) vertebra operated on. 第一個（最上面的）椎骨手術的數量。

# grow tree   
fit <- rpart(Kyphosis ~ Age + Number + Start,  
   method="class", data=kyphosis)  
  
printcp(fit) # display the results

**Classification tree:**

**rpart(formula = Kyphosis ~ Age + Number + Start, data = kyphosis, method = "class")**

**Variables actually used in tree construction:**

**[1] Age Start**

**Root node error: 17/81 = 0.20988**

**n= 81**

complexity param

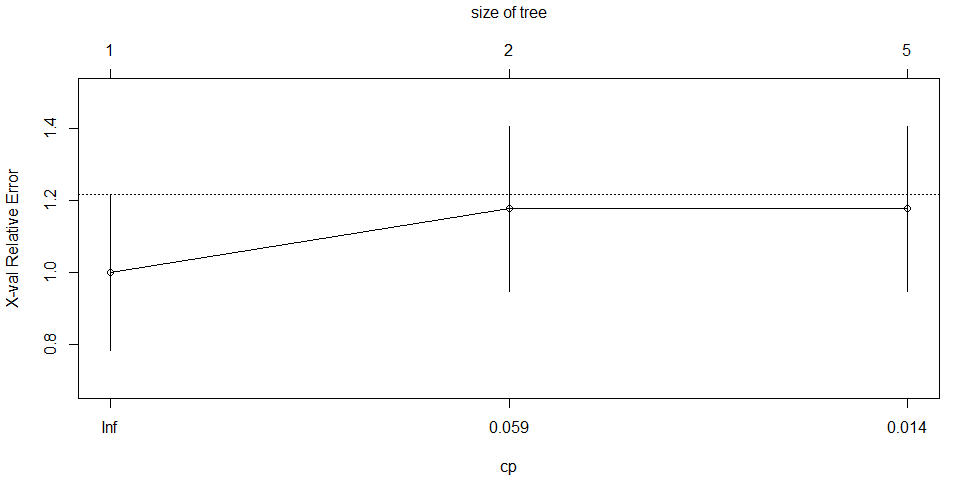
**CP nsplit rel error xerror xstd**

**1 0.176471 0 1.00000 1.0000 0.21559**

**2 0.019608 1 0.82353 1.1765 0.22829**

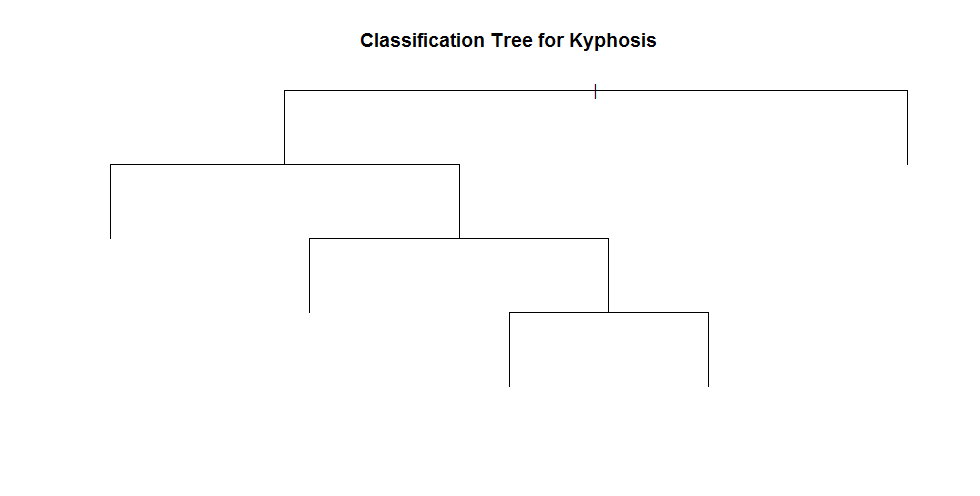
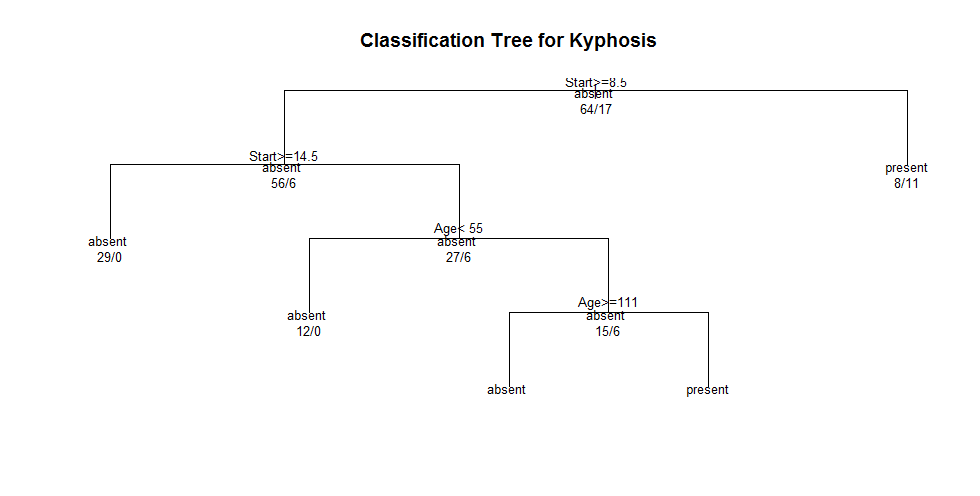
**3 0.010000 4 0.76471 1.1765 0.22829**

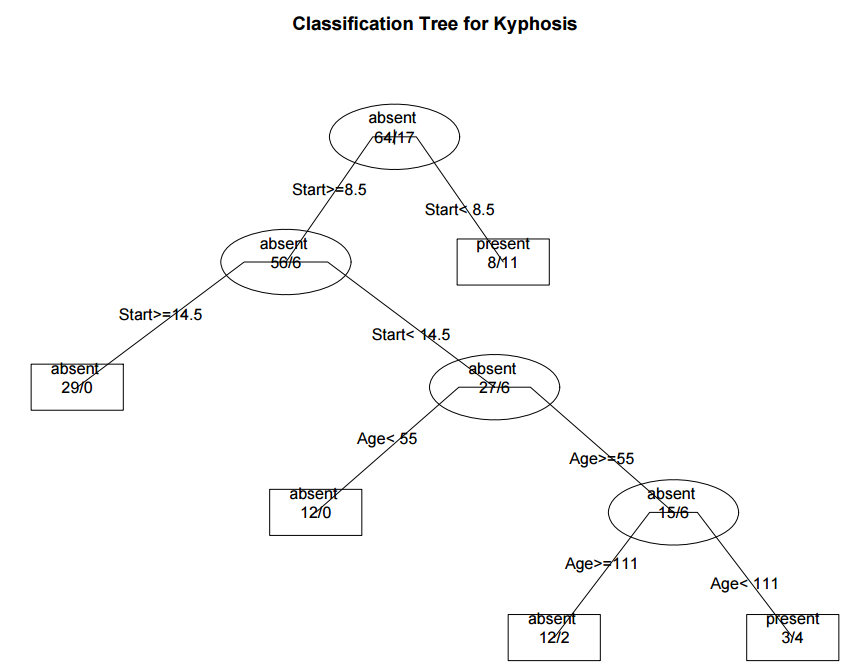
plotcp(fit) # visualize cross-validation results

  
summary(fit) # detailed summary of splits

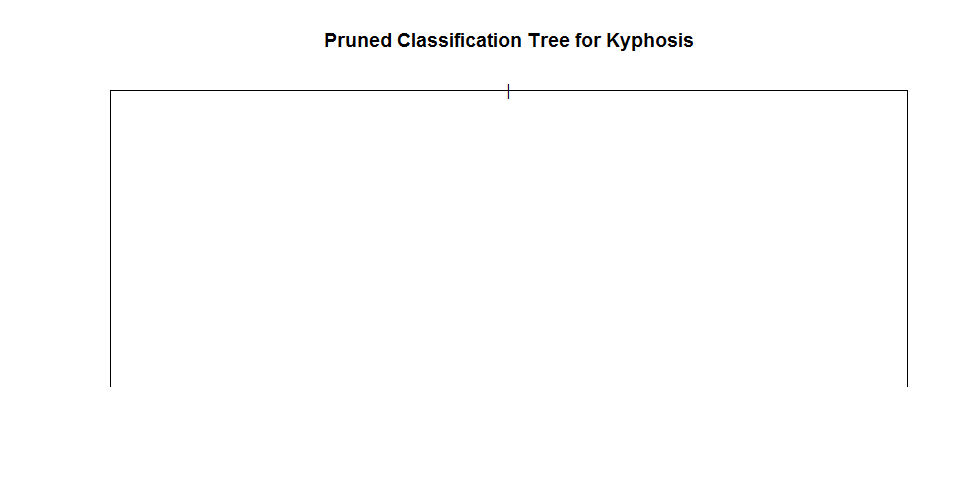
|  |
| --- |
| Call:  rpart(formula = Kyphosis ~ Age + Number + Start, data = kyphosis,  method = "class")  n= 81  CP nsplit rel error xerror xstd  1 0.17647059 0 1.0000000 1.000000 0.2155872  2 0.01960784 1 0.8235294 1.176471 0.2282908  3 0.01000000 4 0.7647059 1.176471 0.2282908  Variable importance  Start Age Number  64 24 12  Node number 1: 81 observations, complexity param=0.1764706  predicted class=absent expected loss=0.2098765 P(node) =1  class counts: 64 17  probabilities: 0.790 0.210  left son=2 (62 obs) right son=3 (19 obs)  Primary splits:主要分離  Start < 8.5 to the right, improve=6.762330, (0 missing)  Number < 5.5 to the left, improve=2.866795, (0 missing)  Age < 39.5 to the left, improve=2.250212, (0 missing)  Surrogate splits:替代分離  Number < 6.5 to the left, agree=0.802, adj=0.158, (0 split)  Node number 2: 62 observations, complexity param=0.01960784  predicted class=absent expected loss=0.09677419 P(node) =0.7654321=62/81  class counts: 56 6  probabilities: 0.903 0.097  left son=4 (29 obs) right son=5 (33 obs)  Primary splits:  Start < 14.5 to the right, improve=1.0205280, (0 missing)  Age < 55 to the left, improve=0.6848635, (0 missing)  Number < 4.5 to the left, improve=0.2975332, (0 missing)  Surrogate splits:  Number < 3.5 to the left, agree=0.645, adj=0.241, (0 split)  Age < 16 to the left, agree=0.597, adj=0.138, (0 split)  Node number 3: 19 observations  predicted class=present expected loss=0.4210526 P(node) =0.2345679=19/81  class counts: 8 11  probabilities: 0.421 0.579  Node number 4: 29 observations  predicted class=absent expected loss=0 P(node) =0.3580247  class counts: 29 0  probabilities: 1.000 0.000  Node number 5: 33 observations, complexity param=0.01960784  predicted class=absent expected loss=0.1818182 P(node) =0.4074074  class counts: 27 6  probabilities: 0.818 0.182  left son=10 (12 obs) right son=11 (21 obs)  Primary splits:  Age < 55 to the left, improve=1.2467530, (0 missing)  Start < 12.5 to the right, improve=0.2887701, (0 missing)  Number < 3.5 to the right, improve=0.1753247, (0 missing)  Surrogate splits:  Start < 9.5 to the left, agree=0.758, adj=0.333, (0 split)  Number < 5.5 to the right, agree=0.697, adj=0.167, (0 split)  Node number 10: 12 observations  predicted class=absent expected loss=0 P(node) =0.1481481  class counts: 12 0  probabilities: 1.000 0.000  Node number 11: 21 observations, complexity param=0.01960784  predicted class=absent expected loss=0.2857143 P(node) =0.2592593  class counts: 15 6  probabilities: 0.714 0.286  left son=22 (14 obs) right son=23 (7 obs)  Primary splits:  Age < 111 to the right, improve=1.71428600, (0 missing)  Start < 12.5 to the right, improve=0.79365080, (0 missing)  Number < 3.5 to the right, improve=0.07142857, (0 missing)  Node number 22: 14 observations  predicted class=absent expected loss=0.1428571 P(node) =0.1728395  class counts: 12 2  probabilities: 0.857 0.143  Node number 23: 7 observations  predicted class=present expected loss=0.4285714 P(node) =0.08641975  class counts: 3 4  probabilities: 0.429 0.571 |

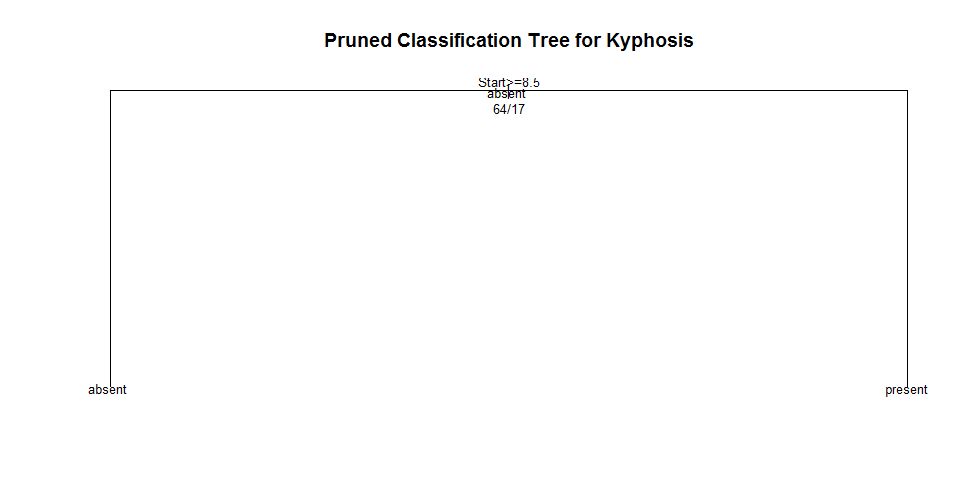
# plot tree   
plot(fit, uniform=TRUE,   
   main="Classification Tree for Kyphosis")

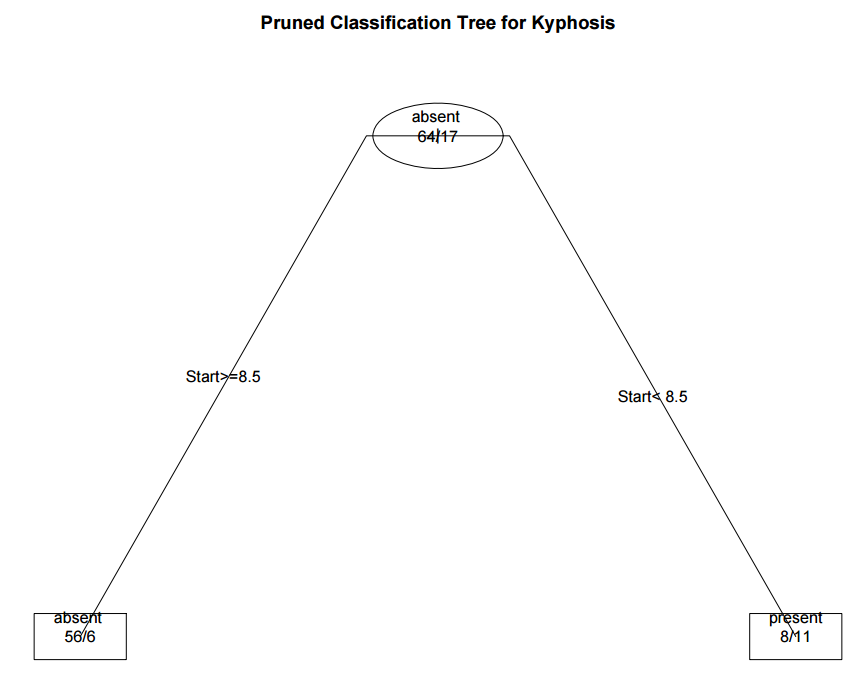
  
text(fit, use.n=TRUE, all=TRUE, cex=.8)  
  
# create attractive postscript plot of tree   
post(fit, file = "c:/tree.ps",   
   title = "Classification Tree for Kyphosis")



# prune the tree   
pfit<-prune(fit, cp=fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"])  
  
# plot the pruned tree   
plot(pfit, uniform=TRUE,   
   main="Pruned Classification Tree for Kyphosis")

  
text(pfit, use.n=TRUE, all=TRUE, cex=.8)

  
post(pfit, file = "c:/ptree.ps",   
   title = "Pruned Classification Tree for Kyphosis")



**Regression Tree example 迴歸樹**

In this example we will predict car mileage from price, country, reliability, and car type. The data frame is **cu.summary**.

# Regression Tree Example  
library(rpart)

Automobile Data from 'Consumer Reports' 1990

**Description**

The cu.summary data frame has 117 rows and 5 columns, giving data on makes of cars taken from the April, 1990 issue of *Consumer Reports*.

Price價格

a numeric vector giving the list price in US dollars of a standard model

Country國家

of origin, a factor with levels

Brazil, England, France, Germany, Japan, Japan/USA, Korea, Mexico, Sweden and USA

Reliability可靠性

an ordered factor with levels

Much worse < worse < average < better < Much better

Mileage里程

fuel consumption miles per US gallon, as tested.

Type

a factor with levels

Compact Large Medium Small Sporty Van

# grow tree   
fit <- rpart(Mileage~Price + Country + Reliability + Type,   
   method="anova", data=cu.summary)  
  
printcp(fit) # display the results

Regression tree:

rpart(formula = Mileage ~ Price + Country + Reliability + Type,

data = cu.summary, method = "anova")

Variables actually used in tree construction:

[1] Price Type

Root node error: 1354.6/60 = 22.576

n=60 (57 observations deleted due to missingness)

CP nsplit rel error xerror xstd

1 0.622885 0 1.00000 1.05589 0.180484

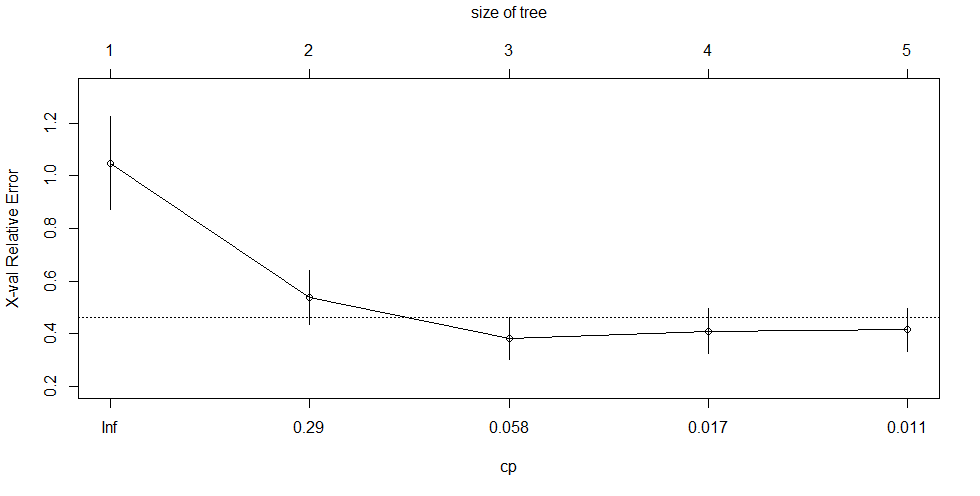
2 0.132061 1 0.37711 0.52725 0.101277

3 0.025441 2 0.24505 0.41337 0.082812

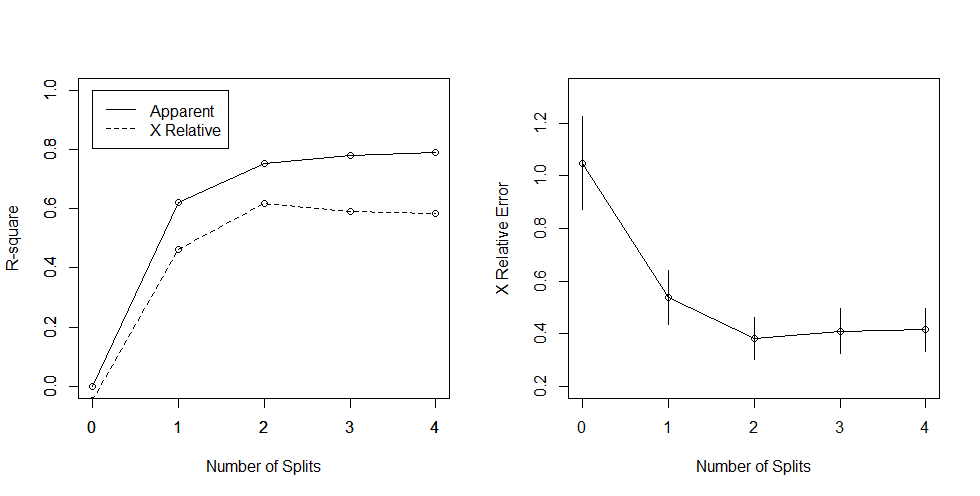
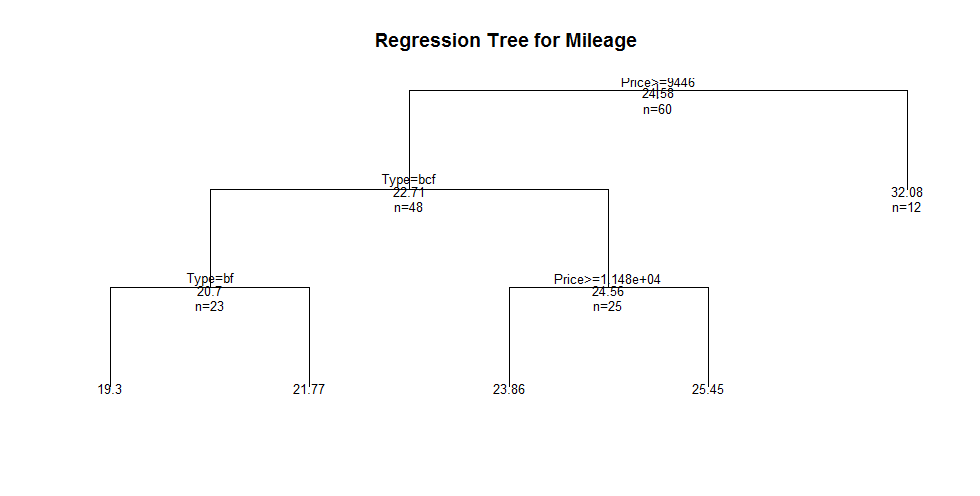
4 0.011604 3 0.21961 0.39568 0.081798

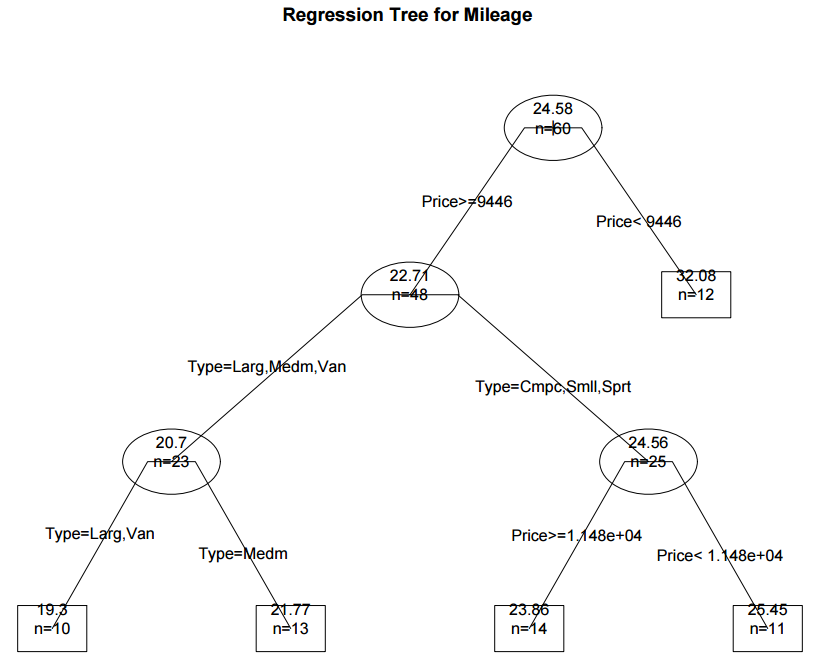
5 0.010000 4 0.20801 0.43176 0.085924

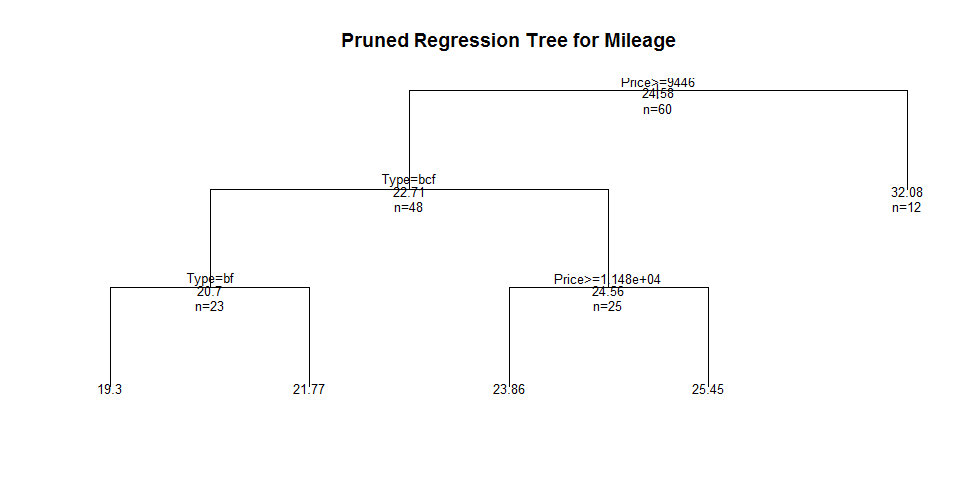
plotcp(fit) # visualize cross-validation results

  
summary(fit) # detailed summary of splits

|  |
| --- |
| Call:  rpart(formula = Mileage ~ Price + Country + Reliability + Type,  data = cu.summary, method = "anova")  n=60 (57 observations deleted due to missingness)  CP nsplit rel error xerror xstd  1 0.62288527 0 1.0000000 1.0558894 0.18048384  2 0.13206061 1 0.3771147 0.5272510 0.10127688  3 0.02544094 2 0.2450541 0.4133746 0.08281166  4 0.01160389 3 0.2196132 0.3956805 0.08179847  5 0.01000000 4 0.2080093 0.4317620 0.08592369  Variable importance  Price Type Country  48 42 10  Node number 1: 60 observations, complexity param=0.6228853  mean=24.58333, MSE=22.57639  left son=2 (48 obs) right son=3 (12 obs)  Primary splits:  Price < 9446.5 to the right, improve=0.6228853, (0 missing)  Type splits as LLLRLL, improve=0.5044405, (0 missing)  Reliability splits as LLLRR, improve=0.1263005, (11 missing)  Country splits as --LRLRRRLL, improve=0.1243525, (0 missing)  Surrogate splits:  Type splits as LLLRLL, agree=0.950, adj=0.750, (0 split)  Country splits as --LLLLRRLL, agree=0.833, adj=0.167, (0 split)  Node number 2: 48 observations, complexity param=0.1320606  mean=22.70833, MSE=8.498264  left son=4 (23 obs) right son=5 (25 obs)  Primary splits:  Type splits as RLLRRL, improve=0.43853830, (0 missing)  Price < 12154.5 to the right, improve=0.25748500, (0 missing)  Country splits as --RRLRL-LL, improve=0.13345700, (0 missing)  Reliability splits as LLLRR, improve=0.01637086, (10 missing)  Surrogate splits:  Price < 12215.5 to the right, agree=0.812, adj=0.609, (0 split)  Country splits as --RRLRL-RL, agree=0.646, adj=0.261, (0 split)  Node number 3: 12 observations  mean=32.08333, MSE=8.576389  Node number 4: 23 observations, complexity param=0.02544094  mean=20.69565, MSE=2.907372  left son=8 (10 obs) right son=9 (13 obs)  Primary splits:  Type splits as -LR--L, improve=0.515359600, (0 missing)  Price < 14962 to the left, improve=0.131259400, (0 missing)  Country splits as ----L-R--R, improve=0.007022107, (0 missing)  Surrogate splits:  Price < 13572 to the right, agree=0.609, adj=0.1, (0 split)  Node number 5: 25 observations, complexity param=0.01160389  mean=24.56, MSE=6.4864  left son=10 (14 obs) right son=11 (11 obs)  Primary splits:  Price < 11484.5 to the right, improve=0.09693168, (0 missing)  Reliability splits as LLRRR, improve=0.07767167, (4 missing)  Type splits as L--RR-, improve=0.04209834, (0 missing)  Country splits as --LRRR--LL, improve=0.02201687, (0 missing)  Surrogate splits:  Country splits as --LLLL--LR, agree=0.80, adj=0.545, (0 split)  Type splits as L--RL-, agree=0.64, adj=0.182, (0 split)  Node number 8: 10 observations  mean=19.3, MSE=2.21  Node number 9: 13 observations  mean=21.76923, MSE=0.7928994  Node number 10: 14 observations  mean=23.85714, MSE=7.693878  Node number 11: 11 observations  mean=25.45455, MSE=3.520661 |

# create additional plots   
par(mfrow=c(1,2)) # two plots on one page   
rsq.rpart(fit) # visualize cross-validation results     
  
# plot tree   
plot(fit, uniform=TRUE,   
   main="Regression Tree for Mileage ")  
text(fit, use.n=TRUE, all=TRUE, cex=.8)  
  
# create attractive postcript plot of tree   
post(fit, file = "c:/tree2.ps",   
   title = "Regression Tree for Mileage ")



# prune the tree   
pfit<- prune(fit, cp=0.01160389) # from cptable     
  
# plot the pruned tree   
plot(pfit, uniform=TRUE,   
   main="Pruned Regression Tree for Mileage")  
text(pfit, use.n=TRUE, all=TRUE, cex=.8)  


post(pfit, file = "c:/ptree2.ps",   
   title = "Pruned Regression Tree for Mileage")

It turns out that this produces the same tree as the original.

**Conditional inference trees via party**

The [**party**](http://cran.r-project.org/web/packages/party/index.html) package provides nonparametric regression trees for nominal, ordinal, numeric, censored, and multivariate responses. [party: A laboratory for recursive partitioning](http://cran.r-project.org/web/packages/party/vignettes/party.pdf), provides details.

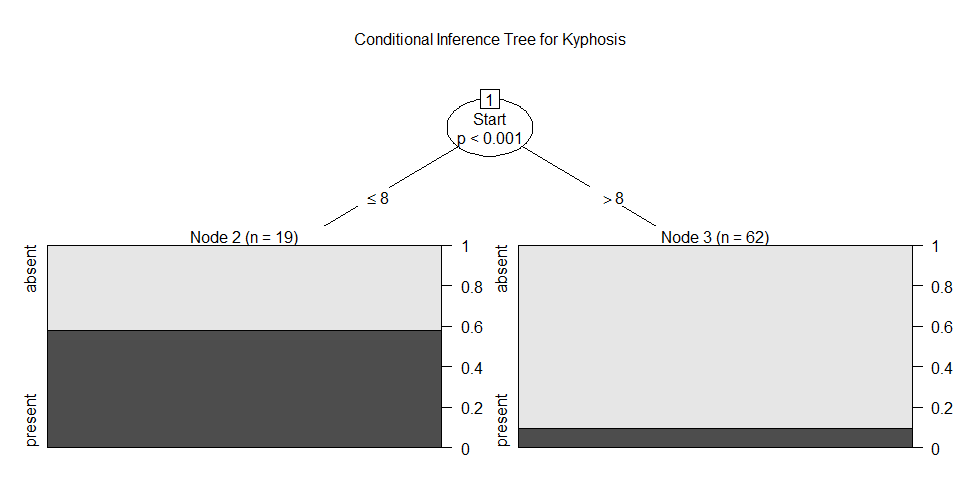
You can create a regression or classification tree via the function

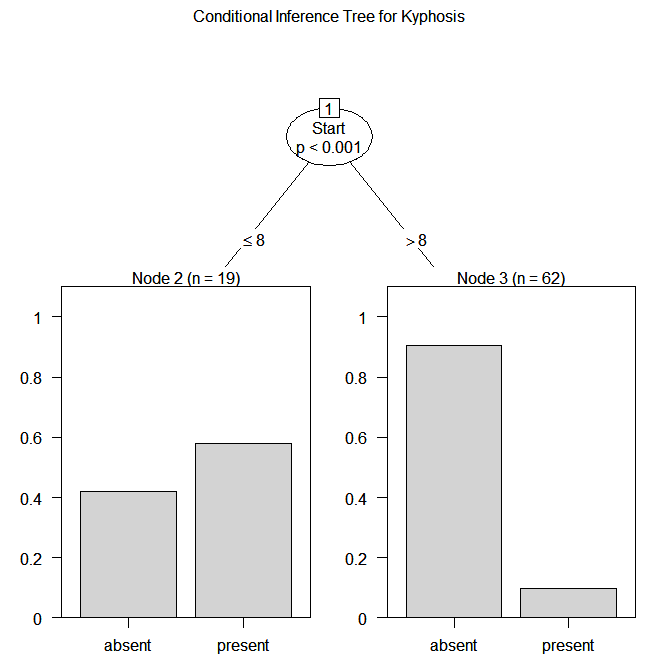
**ctree(***formula*, **data=)**

The type of tree created will depend on the outcome variable (nominal factor, ordered factor, numeric, etc.). Tree growth is based on statistical stopping rules, so pruning should not be required.

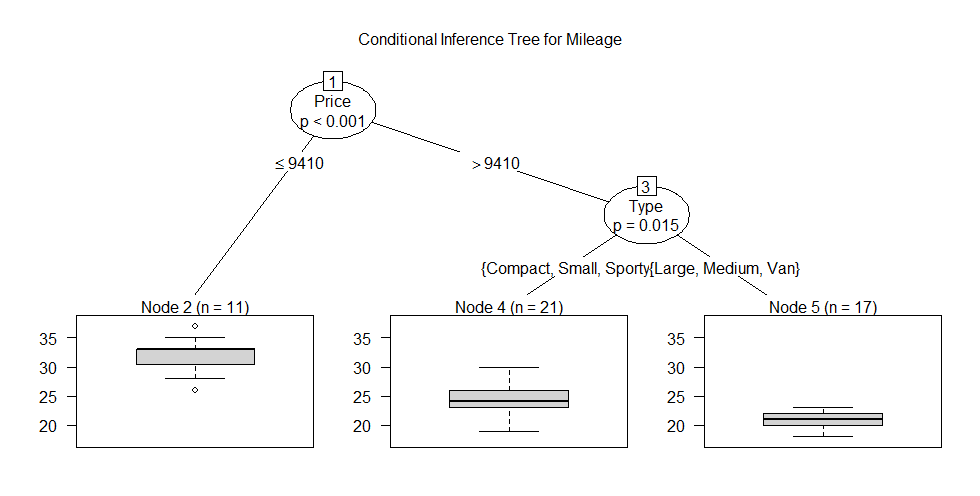
The previous two examples are re-analyzed below.

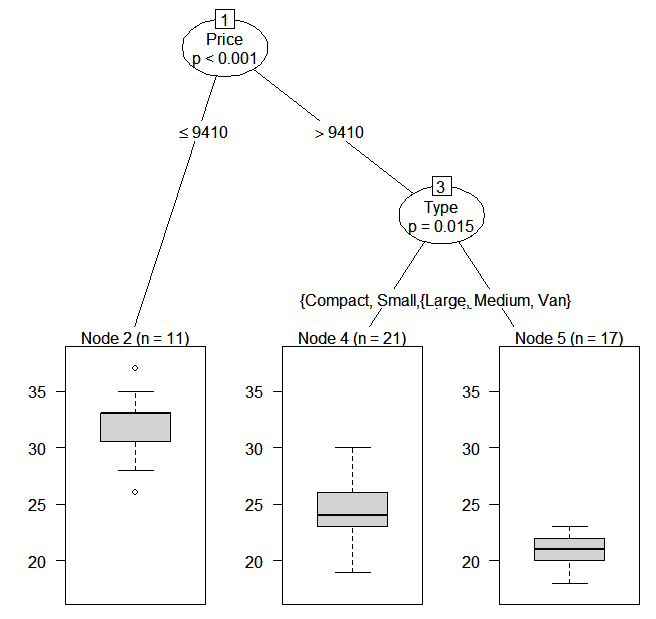
# Conditional Inference Tree for Kyphosis  
library(party)  
fit <- ctree(Kyphosis ~ Age + Number + Start,   
   data=kyphosis)  
plot(fit, main="Conditional Inference Tree for Kyphosis")





# Conditional Inference Tree for Mileage  
library(party)  
fit2 <- ctree(Mileage~Price + Country + Reliability + Type,   
   data=na.omit(cu.summary))



****

Conditional Inference Trees

Conditional inference trees estimate a regression relationship by binary recursive partitioning in a conditional inference framework. Roughly, the algorithm works as follows:

1) Test the global null hypothesis of independence between any of the input variables and the response (which may be multivariate as well). Stop if this hypothesis cannot be rejected. Otherwise select the input variable with strongest association to the response. This association is measured by a p-value corresponding to a test for the partial null hypothesis of a single input variable and the response.

2) Implement a binary split in the selected input variable.

3) Recursively repeats steps 1) and 2).

**Random Forests**

Random forests improve predictive accuracy by generating a large number of bootstrapped trees (based on random samples of variables), classifying a case using each tree in this new "forest", and deciding a final predicted outcome by combining the results across all of the trees (an average in regression, a majority vote in classification). Breiman and Cutler's random forest approach is implimented via the[**randomForest**](http://cran.r-project.org/web/packages/randomForest/index.html) package.

Here is an example.

# Random Forest prediction of Kyphosis data  
library(randomForest)  
fit <- randomForest(Kyphosis ~ Age + Number + Start,   data=kyphosis)  
print(fit) # view results   
importance(fit) # importance of each predictor

For more details see the comprehensive [Random Forest website](http://stat-www.berkeley.edu/users/breiman/RandomForests/).

**Going Further**

This section has only touched on the options available. To learn more, see the CRAN Task View on[Machine & Statistical Learning](http://cran.r-project.org/web/views/MachineLearning.html).