Classification and Regression Trees (CART) with rpart and rpart.plot

Min Ma

August 27, 2014

Titanic Data

The data has 1046 observations on 6 variables.

- `pclass`: passenger class
- `survived`: died or survived
- `sex`: male or female
- `age`: age in years
- `sibsp`: number of siblings or spouses aboard
- `parch`: number of parents or children aboard

```
library(rpart)
library(rpart.plot)
data(ptitanic)

str(ptitanic)
```

```
## 'data.frame':    1309 obs. of  6 variables:
##  $ pclass  : Factor w/ 3 levels "1st","2nd","3rd": 1 1 1 1 1 1 1 1 1 1 ...
##  $ survived: Factor w/ 2 levels "died","survived": 2 2 1 1 1 2 2 1 2 1 ...
##  $ sex     : Factor w/ 2 levels "female","male": 1 2 1 2 1 2 1 2 1 2 ...
##  $ age     :Class 'labelled'  atomic [1:1309] 29 0.917 2 30 25 ...
##   .. ..- attr(*, "units")= chr "Year"
##   .. ..- attr(*, "label")= chr "Age"
##  $ sibsp   :Class 'labelled'  atomic [1:1309] 0 1 1 1 1 0 1 0 2 0 ...
##   .. ..- attr(*, "label")= chr "Number of Siblings/Spouses Aboard"
##  $ parch   :Class 'labelled'  atomic [1:1309] 0 2 2 2 2 0 0 0 0 0 ...
##   .. ..- attr(*, "label")= chr "Number of Parents/Children Aboard"
```

CART Modeling

Make sure all the categorical variables are converted into factors. The function `rpart` will run a regression tree if the response variable is numeric, and a classification tree if it is a factor. See here (http://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf) for a detailed introduction on tree-

based modeling with *rpart* package.

```
# Step1: Begin with a small cp.
set.seed(123)
tree <- rpart(survived ~ ., data = ptitanic, control = rpart.control(cp = 0.0001))

# Step2: Pick the tree size that minimizes misclassification rate (i.e. prediction error
).
# Prediction error rate in training data = Root node error * rel error * 100%
# Prediction error rate in cross-validation = Root node error * xerror * 100%
# Hence we want the cp value (with a simpler tree) that minimizes the xerror.
printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = survived ~ ., data = ptitanic, control = rpart.control(cp = 1e-04))
##
## Variables actually used in tree construction:
## [1] age    parch  pclass sex    sibsp
##
## Root node error: 500/1309 = 0.38
##
## n= 1309
##
##       CP nsplit rel error xerror  xstd
## 1 0.4240      0      1.00   1.00 0.035
## 2 0.0210      1      0.58   0.58 0.030
## 3 0.0150      3      0.53   0.57 0.030
## 4 0.0113      5      0.50   0.57 0.030
## 5 0.0026      9      0.46   0.53 0.029
## 6 0.0020     16      0.44   0.53 0.029
## 7 0.0001     18      0.44   0.53 0.029
```

```
bestcp <- tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]

# Step3: Prune the tree using the best cp.
tree.pruned <- prune(tree, cp = bestcp)
```

The tree has a misclassification rate of `0.38197 * 0.530 * 100% = 20.2%` in cross-validation (i.e. 79.8% of prediction accuracy). Now the pruned tree can be used to produce confusion matrices and tree structure plots.

```
# confusion matrix (training data)
conf.matrix <- table(ptitanic$survived, predict(tree.pruned,type="class"))
rownames(conf.matrix) <- paste("Actual", rownames(conf.matrix), sep = ":")
colnames(conf.matrix) <- paste("Pred", colnames(conf.matrix), sep = ":")
print(conf.matrix)
```

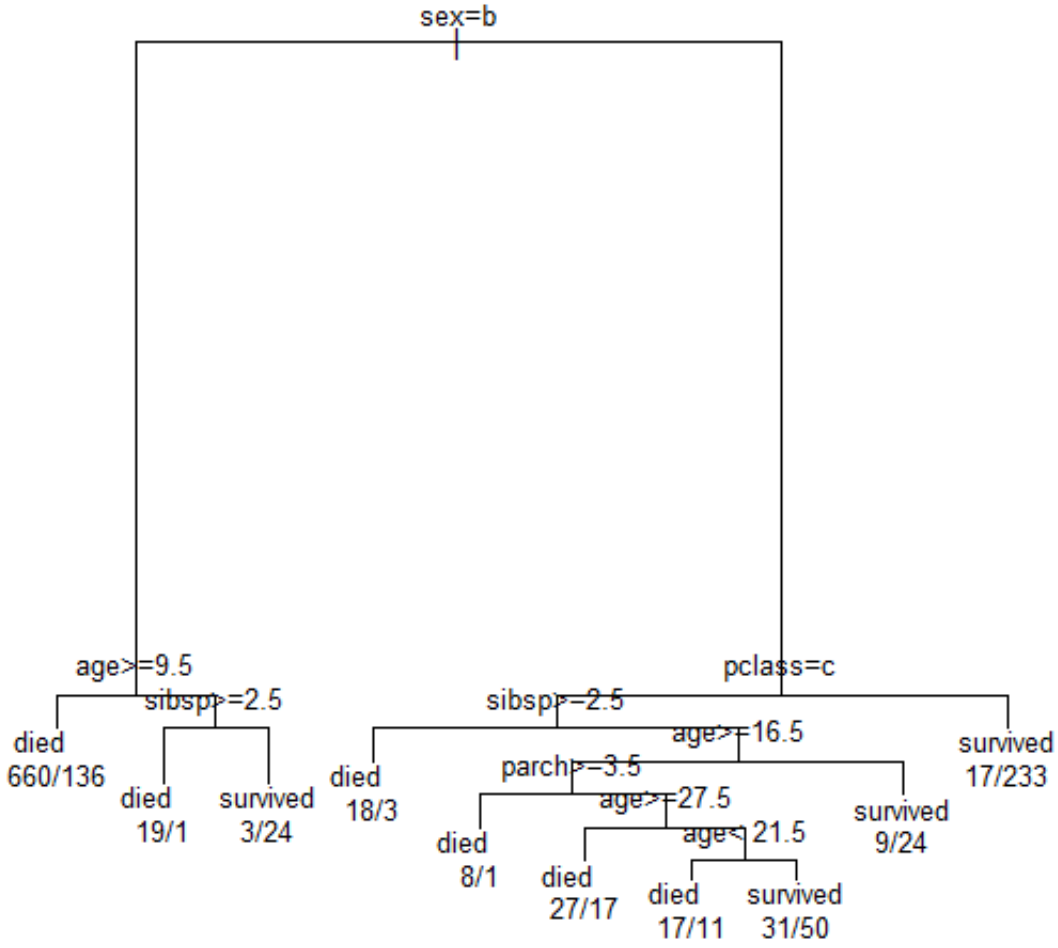
```
##
##                    Pred:died Pred:survived
##   Actual:died            749             60
##   Actual:survived        169            331
```

Plots

Simple plots can be generated using `plot.rpart` and `text.rpart` functions in *rpart* package. They can be quite ugly and hard to read, especially when you have many levels for a factor since the plot automatically labels them using alphabets.

```
plot(tree.pruned)
text(tree.pruned, cex = 0.8, use.n = TRUE, xpd = TRUE)
```

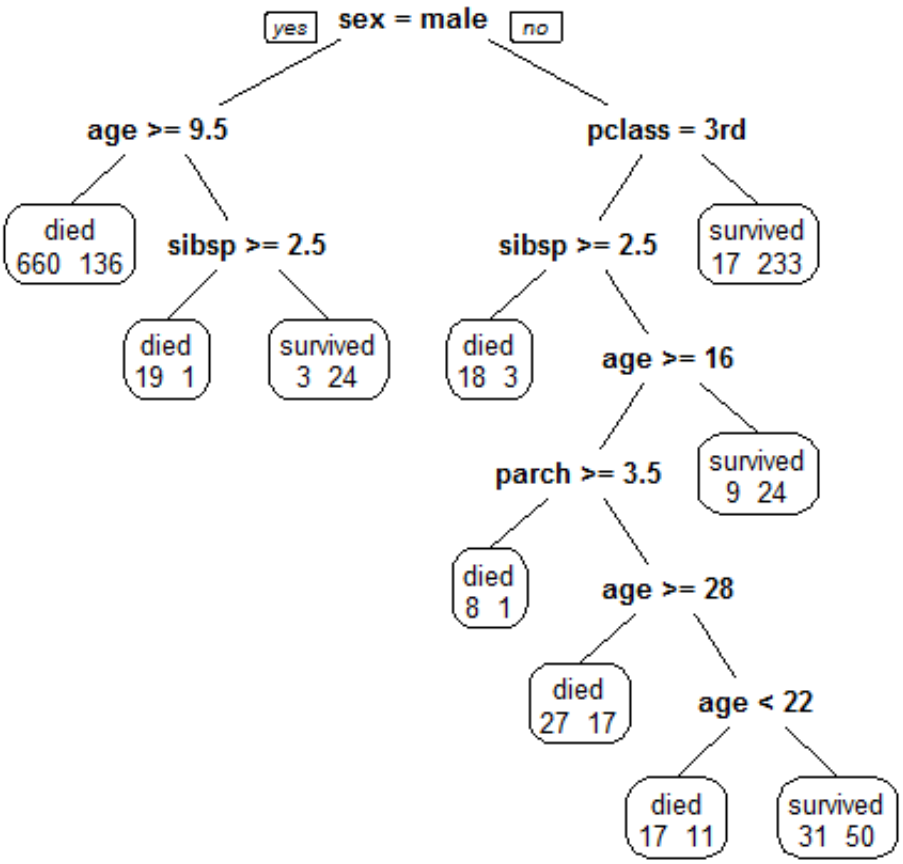
```
# use.n = TRUE adds number of observations at each node
# xpd = TRUE keeps the labels from exteding outside the plot
```



Take the far left node as an example, 660/136 under “died” means 660 people that actually died and 136 that actually survived are predicted as died. A similar but better-looking plot can be produced using `prp` function in *rpart.plot* package.

```
prp(tree.pruned, faclen = 0, cex = 0.8, extra = 1)
```

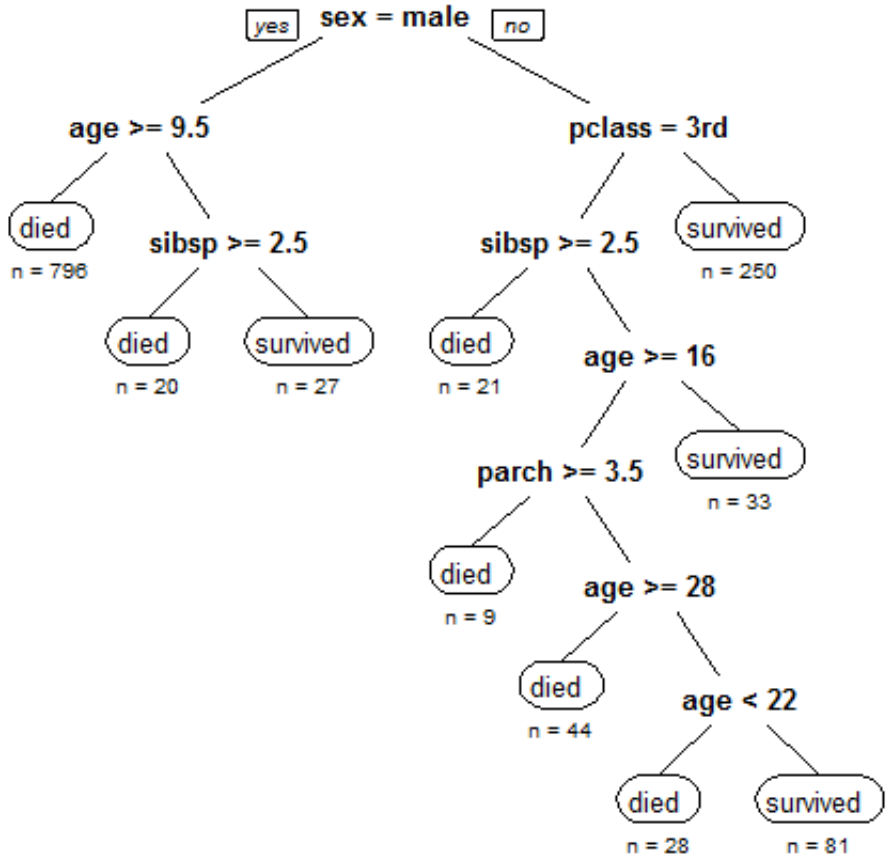
```
# faclen = 0 means to use full names of the factor labels
# extra = 1 adds number of observations at each node; equivalent to using use.n = TRUE i
n plot.rpart
```



A very detailed tutorial on how to make fancy tree plots with *rpart.plot* can be found here (http://www.milbo.org/rpart-plot/prp.pdf). It helped me to figure out a way to add a total count at each node instead of separate counts from different actual groups (i.e. I wanted n = 796 instead of 660/136). The following function `tot_count` is modified based on the `node.fun3` in the tutorial.

```
tot_count <- function(x, labs, digits, varlen)
{
  paste(labs, "\n\nn =", x$frame$n)
}

prp(tree.pruned, faclen = 0, cex = 0.8, node.fun=tot_count)
```

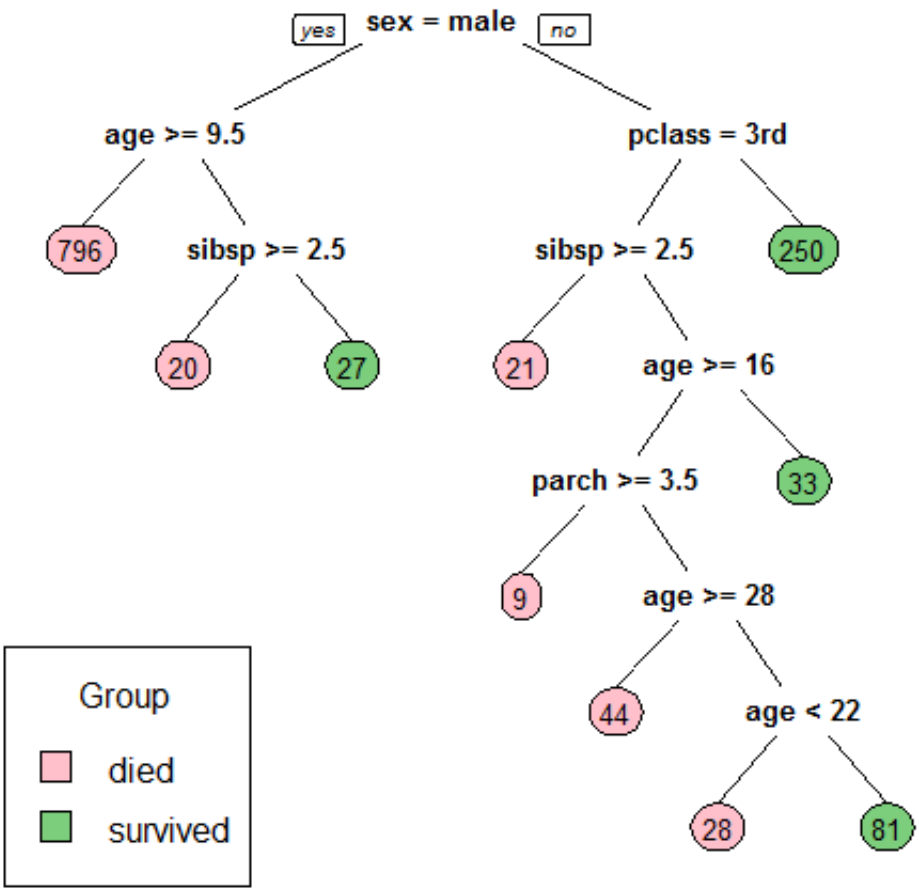


Sometimes if you have a big tree then the plot might look messy and is not very readable. It happened to me when I was plotting a tree with over 30 nodes. It might be helpful to use different colors to denote the predicted categories and only put the counts within the circles. Again this method was inspired by the tutorial from Stephen Milborrow (http://www.milbo.users.sonic.net/).

```
only_count <- function(x, labs, digits, varlen)
{
  paste(x$frame$n)
}

boxcols <- c("pink", "palegreen3")[tree.pruned$frame$yval]

par(xpd=TRUE)
prp(tree.pruned, faclen = 0, cex = 0.8, node.fun=only_count, box.col = boxcols)
legend("bottomleft", legend = c("died","survived"), fill = c("pink", "palegreen3"),
       title = "Group")
```


yes
sex = male
no
age >= 9.5
pclass = 3rd
796
sibsp >= 2.5
sibsp >= 2.5
250
20
27
21
age >= 16
parch >= 3.5
33
9
age >= 28
44
age < 22
28
81
Group
died
survived