Classification and regression trees

Recursive partitioning methods provide an alternative to (generalized) linear models for categorical responses, particularly when there are numerous potential predictors and/or there are important interactions among predictors. These methods attempt to define a set of rules to classify observations into mutually exclusive subsets based on combinations of the explanatory variables, and tend to work well when there are important non-linearities or interactions in the data.

We illustrate some of the ideas behind these methods with the rpart package, for fitting recursive partitioning trees,[1] and the party package, which embeds tree-structured models within a general framework of conditional inference procedures.

For a binary response variable, `rpart()` fits a tree to predict the outcome using a two-stage procedure: First, the observations are partitioned into prediction classes (e.g., ”lived”, ”died”) by using univariate, binary splits on the available predictors in a recursive way. A second stage is then applied to evaluate the resulting binary tree, using some methods of testing and cross-validation to prune the tree according to some criterion.

The tree is built by the following process: First the single variable is found which best splits the data into two groups. The data is separated, and then this process is applied separately to each sub-group, and so on recursively until the subgroups either reach a minimum size or until no improvement can be made.

Example: Recursive partitioning trees [titanic-tree]

This example uses the `Titanicp` data we have examined in other examples. We first fit an `rpart()` tree to predict survival using only age and passenger class. The rpart.plot package can plot such trees, and provides numerous options to control the details of what is plotted for each node. The focus here is on visualization methods for interpreting the results.

```
# fit a simple tree, using only pclass and age
library(rpart)
library(rpart.plot)
data(Titanicp, package="vcdExtra")
rp0 <- rpart(survived ~ pclass + age, data=Titanicp)
rpart.plot(rp0, type=0, extra=2, cex=1.5)
```

The basic tree shown in Figure 1 displays the levels of the variables used for classification at each node. The tree is read as follows: Each non-terminal (unboxed) node represents a decision based on one variable, where the left branch corresponds to `TRUE` and the right branch `FALSE`.

For example, the first split separates 3rd class passengers from the rest, and the next split on the right is between those older than 16, in class 1 and 2. The terminal nodes (leaves) with ovals indicate the prediction for that partition and number who lived or died out of the total in that subgroup. For example, those with `age<16` in class 1 and 2 are predicted to have survived and 34 out of 36 did so. The bottom left leaf represents 1st class passengers with `age<60`, where 187 out of 294 survived.

Printing the resulting "rpart" object (`rp0`, here) gives a text representation of the model. The legend indicates that each line contains the node number, splitting variable and value (`split`), the number (`n`) of observations at that node, the number (`loss`) of observations incorrectly classified, the predicted classification at that node (`yval`), and the probabilities of the response classes at that node (`yprob`). Leaf nodes are indicated by the trailing `"*"`.

```
rp0

## n= 1309
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
```

[1] rpart is an R implementation of the CART (*Classification and Regression Trees*) book and software of Breiman *et al.* (1984)

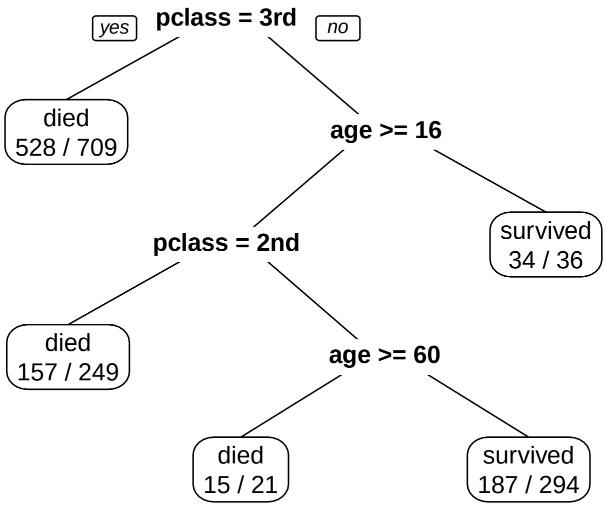

Figure 1: Classification tree for passengers on the Titanic, using *pclass* and *age*

```
##  1) root 1309 500 died (0.61803 0.38197)
##    2) pclass=3rd 709 181 died (0.74471 0.25529) *
##    3) pclass=1st,2nd 600 281 survived (0.46833 0.53167)
##      6) age>=15.5 564 279 survived (0.49468 0.50532)
##       12) pclass=2nd 249  92 died (0.63052 0.36948) *
##       13) pclass=1st 315 122 survived (0.38730 0.61270)
##         26) age>=60.5 21   6 died (0.71429 0.28571) *
##         27) age< 60.5 294 107 survived (0.36395 0.63605) *
##      7) age< 15.5 36   2 survived (0.05556 0.94444) *
```

The information here is a faithful textual representation of the tree, but much harder to read than the visual representation in Figure 1.

Any such recursive partitioning tree can be visualized instead by a ***treemap*** or partition map, which divides a unit rectangle into regions based on the variable splits.[2] This is easiest to show for two variables, where we can also plot the individual observations. Figure 2 is the treemap representation of the tree in Figure 1. The R code uses basic `plot()` facilities and isn't shown here.

A similar treemap can be produced using the plotmo package. This has many options for plotting a model response in models with one or two predictors. In the call below, `type2="image"` says to plot the two-way effect of `pclass:age` as a shaded image.

```r
library(plotmo)
plotmo(rp0, nresponse="survived", degree1=0, type2="image",
  col.image=gray(seq(.6, 1,.05)),
  col.response=ifelse(Titanicp$survived=="died", "red", "blue"),
  pch=ifelse(Titanicp$survived=="died", 15, 16))
```

`plotmo()` is much more general than this, and can handle models fit with `lm()`, `glm()`, `gam()`, `lda()` and others in addition to `rpart()`. Similar to an effects plot, it allows plotting the response in a model as one or two predictors are varied, holding all other predictors constant (continuous variables at their medians; factors, at their first level, by default). By default, the function plots the response for all main

[2] The idea for this plot comes from Varian (2013), `http://people.ischool.berkeley.edu/~hal/Papers/2013/ml.pdf`.

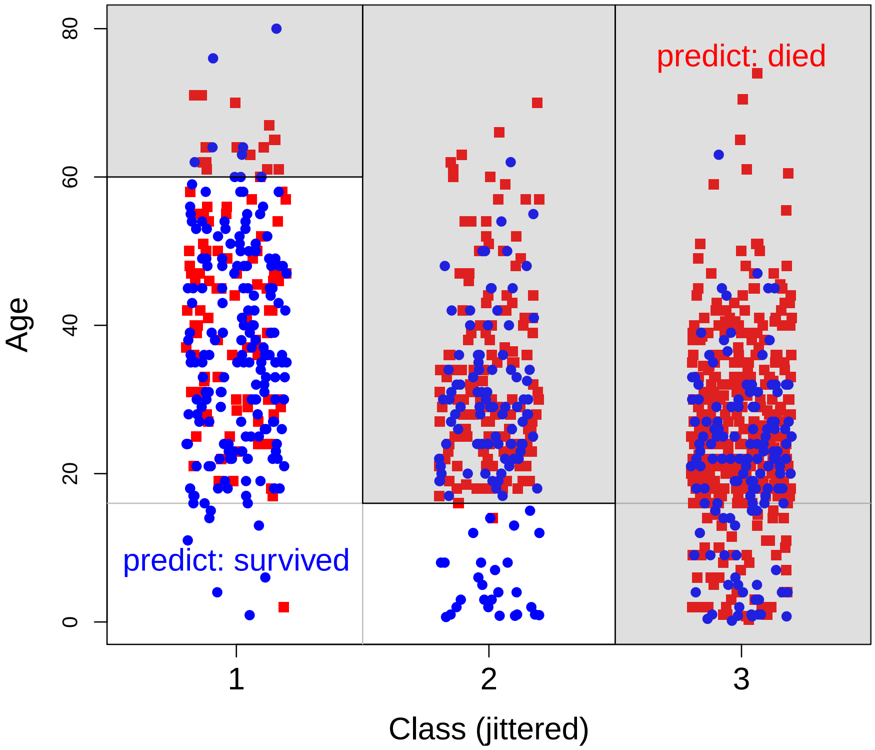

Figure 2: Partition map for the tree in Figure 1. Those in the shaded region are predicted to have died; observations are shown by red circles (died) and blue squares (survived)

effects (`degree1=`) and all two-way effects (`degree2=`) in a single multi-panel plot. Here we produce them separately, in order to lay them out side-by-side.

```
# one-way plots
plotmo(rp0, nresponse="survived", degree1=1, degree2=0, trace=-1, do.par=FALSE)
plotmo(rp0, nresponse="survived", degree1=2, degree2=0, trace=-1, do.par=FALSE)
# two-way, 3D persp plot
plotmo(rp0, nresponse="survived", degree1=0, trace=-1)
```

Note that the image plot in Figure 3 is a view from above of the 3D right panel in Figure 4. Some care is needed in interpreting the one-way plots When there are factors in the model, because the default "constant" value is the first level of a factor. For example, the middle plot shows the predictions for *age* for those in 1^{st} class.[3] This can be seen in as the profile of the right-most step function in the 3D image i the right panels Figure 3.

Pruning: At each split, `rpart()` calculates a number of statistics, including a complexity parameter (`cp`) and measures of the error (`error`) in classification, as well as the mean (`xerror`) and standard deviation (`xstd`) of the errors in the cross-validated prediction. This information can be printed using `printcp()` and plotted with `printcp()`, as shown in Figure 5.

```
printcp(rp0)

##
## Classification tree:
## rpart(formula = survived ~ pclass + age, data = Titanicp)
```

[3] Effect plots, in the effects package provide a more general way to average over factors, but are not available for "rpart" models

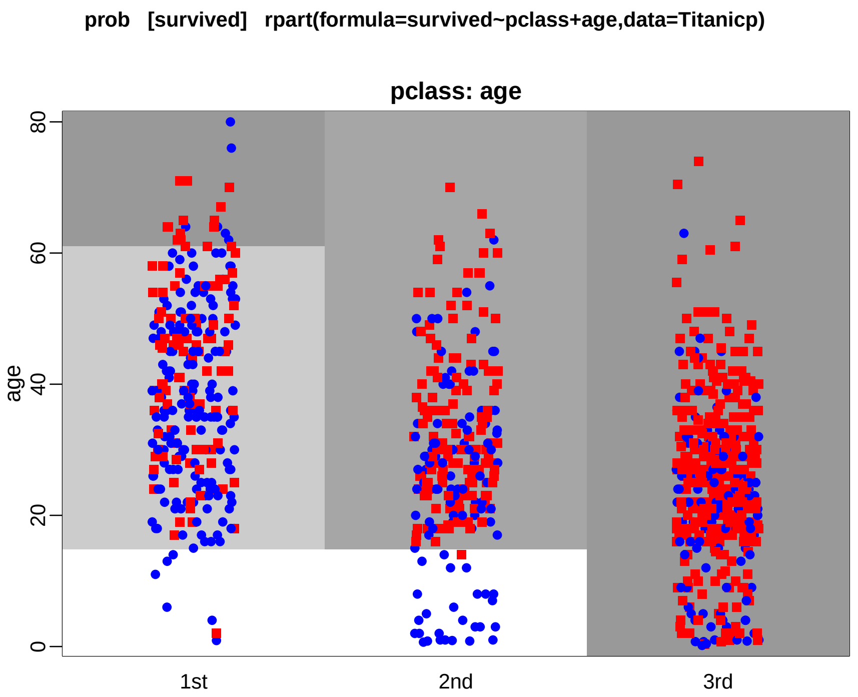

Figure 3: `plotmo()` plot for the tree in Figure 1. Shading level is proportional to the predicted probability of survival.

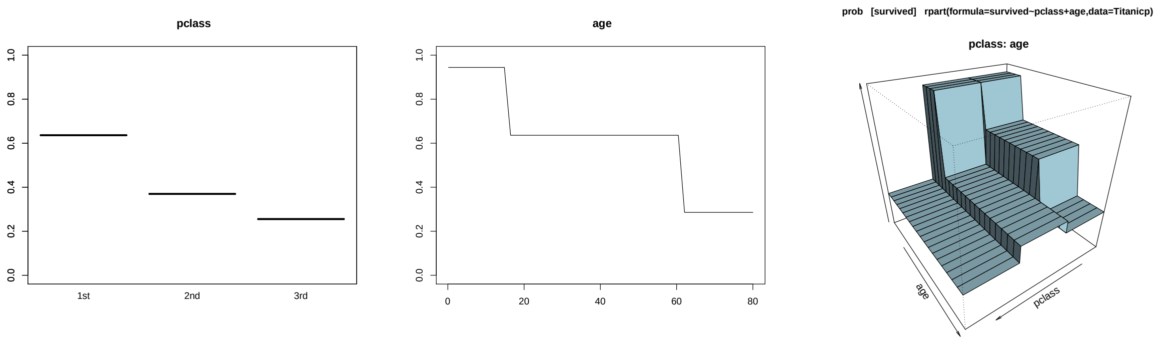

Figure 4: Other `plotmo()` plots: one-way and two-way effects

```
##
## Variables actually used in tree construction:
## [1] age    pclass
##
## Root node error: 500/1309 = 0.38
##
## n= 1309
##
##      CP nsplit rel error xerror  xstd
## 1 0.076      0      1.00   1.00 0.035
## 2 0.065      1      0.92   0.96 0.035
## 3 0.018      3      0.79   0.83 0.034
## 4 0.010      4      0.78   0.82 0.034
```

```
plotcp(rp0, lty=2, col="red", lwd=2)
```

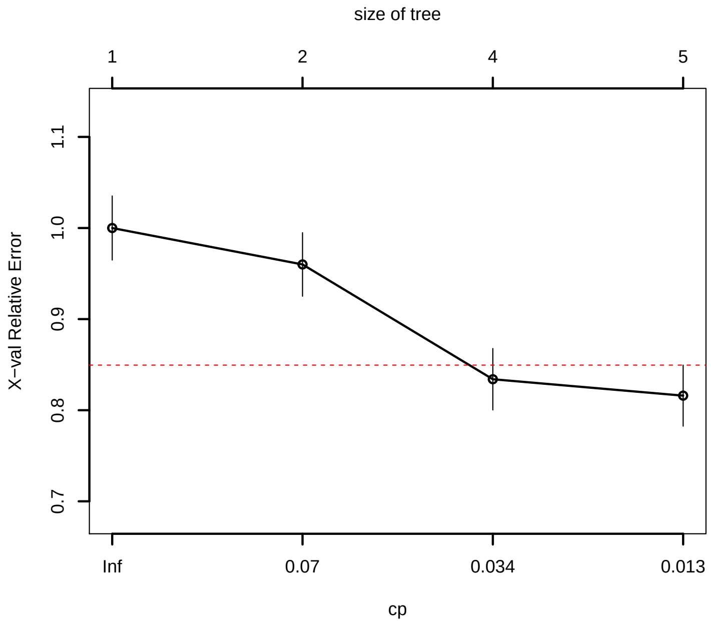

Figure 5: Plot of complexity and error statistics. The dashed horizontal line is drawn 1 SE above the minimum of the curve.

There is not really any need to prune this small tree, but for illustration, we can use this information to prune the tree, and then plot the pruned result, this time with additional options (Figure 6). The option `extra=2` shows the classification of the observations that reach each node and `box.col` allows the node ovals to be colored.

```
rp0.pruned <- prune(rp0, cp=.05)
rpart.plot(rp0.pruned, type=0, extra=2, cex=1.5,
           under=TRUE, box.col=c("pink", "lightblue")[rp0.pruned$frame$yval])
```

Larger models: Continuing, we now include ***sex*** and ***sibsp*** (number of siblings and parents) among the predictors.

```
rp1 = rpart(survived ~ pclass + sex + age + sibsp, data=Titanicp)
```

In the table of complexity parameters, there is no evidence of a need to prune this tree.

```
printcp(rp1)

##
## Classification tree:
## rpart(formula = survived ~ pclass + sex + age + sibsp, data = Titanicp)
##
## Variables actually used in tree construction:
## [1] age    pclass sex    sibsp
```

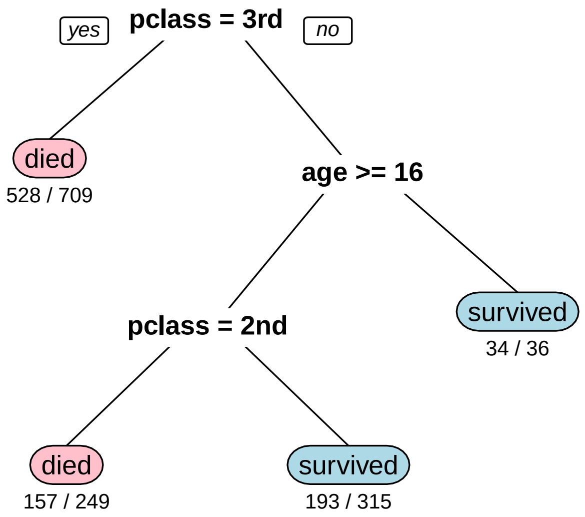

Figure 6: Classification tree for passengers on the Titanic, pruned

```
##
## Root node error: 500/1309 = 0.38
##
## n= 1309
##
##      CP nsplit rel error xerror  xstd
## 1 0.424      0      1.00   1.00 0.035
## 2 0.021      1      0.58   0.58 0.030
## 3 0.015      3      0.53   0.56 0.030
## 4 0.014      5      0.50   0.55 0.029
## 5 0.012      7      0.48   0.55 0.029
## 6 0.010      8      0.46   0.53 0.029
```

We plot the tree, using the `rpart.plot()` again, giving Figure 7.

```
rpart.plot(rp1, type=4, extra=2, faclen=0, under=TRUE, cex=1.1,
  box.col=c("pink", "lightblue")[rp1$frame$yval])
```

sex is now the primary splitting variables. Interestingly, among male passengers, with an overall low survival rate, survival is next predicted by *age* and then family size (*sibsp*); passenger class doesn't matter here. Among female passengers, those in class 1 and 2 are predicted to survive, regardless of other predictors, while predictions for those in 3^{rd} class with large families (`sibsp` *ge* 2.5) are gloomy ("died"), but among those in smaller predicted outcome depends on age.

This example nicely illustrates the difference between tree-based models and standard (generalized) linear models for a binary response. Recursive partitioning methods provide a nested set of decision rules depending on various subsets of the predictor variable values. There is no need, within these methods to ask or test whether the predictor effects are linear or nonlinear, nor is there an need to model interactions among predictors explicitly (which ones?), because the recursive partioning of the observations automatically takes care of combinations of predictors that appear in the various branches.

△

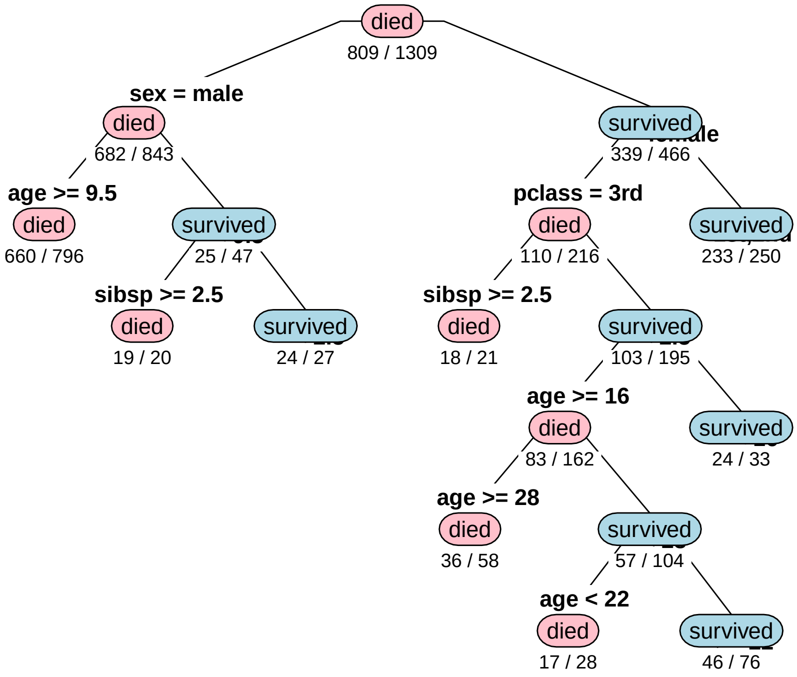

Figure 7: Plot of the extended `rpart()` tree for four predictors

Example: Conditional inference trees [titanic-ctree]

`rpart()` classification trees use internal cross-validation to balance model complexity against goodness of fit, particularly for out-of-sample prediction. The complexity parameter (`cp`) mentioned earlier imposes a cost for having many branches, in a way analogous to other shrinkage methods (lasso, ridge regression, etc.) and statistics (AIC, BIC).

However, these methods are subject to overfitting (hence, the need for pruning) and have an ad hoc flavor, since they don't use any statistical notion to distinguish significant from insignificant improvements with additional splits. The conditional inference framework embodied in the party package solves these problems by embedding recursive partioning methods within a general theory of permutation tests stemming from Strasser and Weber (1999)

Essentially, a linear statistic is calculated to test the hypothesis of independence between the response, Y, and each predictor. A p-value for that test can be calculated with reference to the permutation distribution over all permutations of Y. The procedure stops if none of the predictors allows this null hypothesis to be rejected; otherwise the variable with the strongest association to Y is selected for the next split.

Here we fit a conditional inference tree, using `ctree()` from the party package and the predictors *pclass*, *sex* and *age*.

```
library(party)
titanic.ctree = ctree(survived ~ pclass + sex + age, data=Titanicp)
titanic.ctree

##
##   Conditional inference tree with 8 terminal nodes
##
## Response:  survived
## Inputs:  pclass, sex, age
## Number of observations:  1309
```

```
##
## 1) sex == {male}; criterion = 1, statistic = 365.607
##   2) pclass == {1st}; criterion = 1, statistic = 32.994
##     3) age <= 54; criterion = 0.992, statistic = 9.079
##       4)*  weights = 151
##     3) age > 54
##       5)*  weights = 28
##   2) pclass == {2nd, 3rd}
##     6) age <= 9; criterion = 1, statistic = 25.406
##       7) pclass == {2nd}; criterion = 0.998, statistic = 12.103
##         8)*  weights = 11
##       7) pclass == {3rd}
##         9)*  weights = 29
##     6) age > 9
##       10)*  weights = 624
## 1) sex == {female}
##   11) pclass == {1st, 2nd}; criterion = 1, statistic = 115.454
##     12) pclass == {2nd}; criterion = 0.956, statistic = 5.911
##       13)*  weights = 106
##     12) pclass == {1st}
##       14)*  weights = 144
##   11) pclass == {3rd}
##     15)*  weights = 216
```

The `plot()` method for "ctree" objects is very flexible. The following call produces Figure 8. Each non-terminal node is labeled with the *p*-value for that split, and the barplot for each leaf node shows the proportion of survivors on that branch. The arguments to `plot.ctree()` can include panel functions used to plot the interior nodes (`inner_panel=node_inner` here) and leaf nodes (`terminal_panel=node_barplot` here).

```
plot(titanic.ctree,
  tp_args = list(fill = c("blue", "lightgray")),
  ip_args = list(fill = c("lightgreen"))
        )
```

△

References

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and regression trees.* California: Wadsworth.

Hothorn, T., Hornik, K., Strobl, C., and Zeileis, A. (2013). *party: A Laboratory for Recursive Partytioning.* URL `http://CRAN.R-project.org/package=party`. R package version 1.0-9.

Milborrow, S. (2012a). *plotmo: Plot a model's response while varying the values of the predictors.* URL `http://CRAN.R-project.org/package=plotmo`. R package version 1.3-2.

Milborrow, S. (2012b). *rpart.plot: Plot rpart models. An enhanced version of plot.rpart.* URL `http://CRAN.R-project.org/package=rpart.plot`. R package version 1.4-3.

Strasser, H. and Weber, C. (1999). On the asymptotic theory of permutation statistics. *Mathematical Methods of Statistics*, 8, 220–250.

Therneau, T. M. and Atkinson, B. (2013). *rpart: Recursive Partitioning.* URL `http://CRAN.R-project.org/package=rpart`. R package version 4.1-3.

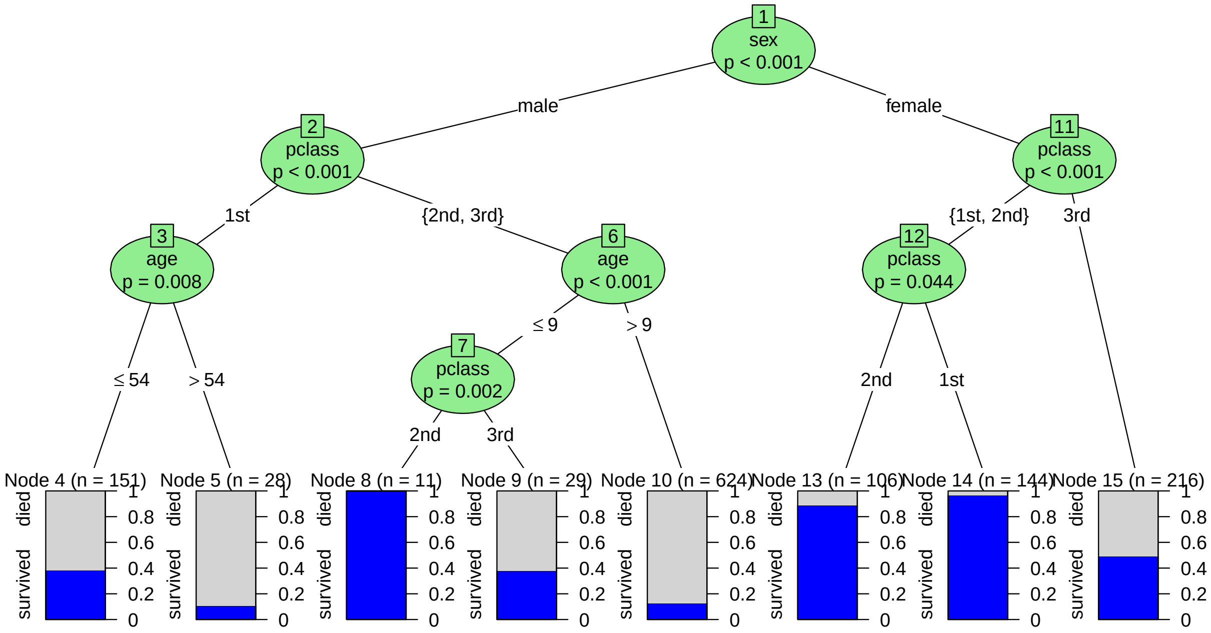

Figure 8: A conditional inference tree for survival on the Titanic. The barplots below each leaf node highlight the proportion of survivors in each branch.

R Packages used: party, modeltools, stats4, strucchange, sandwich, zoo, grid, plotmo, rpart.plot, rpart.