University of Zurich[UZH]

IFSPM Institut für Sozial- und Präventivmedizin

Constant Partying: Introducing a Generic Toolkit for Recursive Partitioning in R

Torsten Hothorn (Universität Zürich)
Achim Zeileis (Universität Innsbruck)

Celebrating 50th anniversary

PROBLEMS IN THE ANALYSIS OF SURVEY DATA, AND A PROPOSAL

JAMES N. MORGAN AND JOHN A. SONQUIST*
University of Michigan

Most of the problems of analyzing survey data have been reasonably well handled, except those revolving around the existence of interaction effects. Indeed, increased efficiency in handling multivariate analyses even with non-numerical variables, has been achieved largely by assuming additivity. An approach to survey data is proposed which imposes no restrictions on interaction effects, focuses on importance in reducing predictive error, operates sequentially, and is independent of the extent of linearity in the classifications or the order in which the explanatory factors are introduced.

JASA **58**(302) June 1963 (well, there were no early online versions, so most people read the paper in 1964 for the first time).

We need a partykit!

Celebrating 50th anniversary

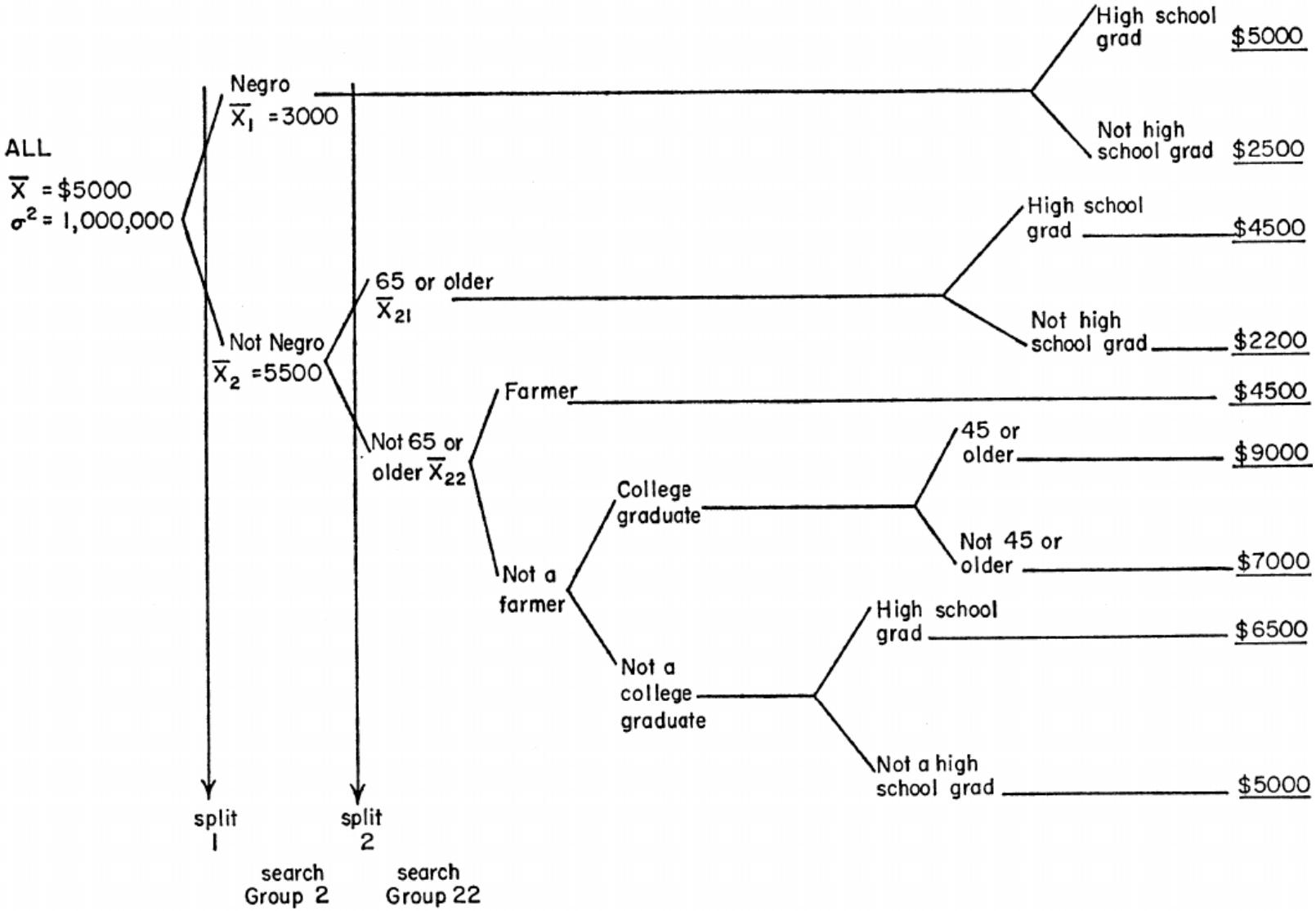


CHART II. Annual Earnings.

Overview

- Status quo: R software for tree models
- New package: partykit
  - Unified infrastructure for recursive partytioning
  - Classes and methods
  - Interfaces to `rpart`, `J48`, ...
  - Illustrations
- Here and now: Intro to partykit for trees with constant fits in each node.

R software for tree models

Status quo: The CRAN task view on "Machine Learning" at http://CRAN.R-project.org/view=MachineLearning lists numerous packages for tree-based modeling and recursive partitioning, including

- rpart (CART),
- tree (CART),
- mvpart (multivariate CART),
- RWeka (J4.8, M5', LMT),
- party (CTree, MOB),
- and many more (C50, quint, stima, ...).

Related: Packages for tree-based ensemble methods such as random forests or boosting, e.g., randomForest, gbm, mboost, etc.

R software for tree models

Furthermore: Tree algorithms/software without R interface, e.g.,

- QUEST,
- GUIDE,
- LOTUS,
- CRUISE,
- ...

Currently: All algorithms/software have to deal with similar problems *but* provide different solutions without reusing code.

R software for tree models

Challenge: For implementing new algorithms in R, code is required not only for fitting the tree model but also

- representing fitted trees,
- printing trees,
- plotting trees,
- computing predictions from trees.

R software for tree models

Question: Wouldn't it be nice if there were an R package that provided code for

- representing fitted trees,
- printing trees,
- plotting trees,
- computing predictions from trees?

R software for tree models

Answer: The R package partykit provides unified infrastructure for recursive partytioning, especially

- representing fitted trees,
- printing trees,
- plotting trees,
- computing predictions from trees!

partykit: Unified infrastructure

Design principles: Toolkit for recursive partytitioning.

- One ‘agnostic’ base class which can encompass an extremely wide range of different types of trees.
- Subclasses for important types of trees, e.g., trees with constant fits in each terminal node.
- Nodes are recursive objects: nodes can contain child nodes.
- Keep data out of the recursive node and split structure.
- Basic print, plot, and predict for raw node structure.
- Customization via panel or panel-generating functions.
- Coercion from existing objects (e.g., rpart) to new class.
- Use simple/fast S3 classes and methods.

partykit: Base classes

Class constructors: Generate basic building blocks.

- `partysplit(varid, breaks = NULL, index = NULL, ...)` where `breaks` provides the breakpoints wrt variable `varid`; `index` determines to which kid node observations are assigned.
- `partynode(id, split = NULL, kids = NULL, ...)` where `split` is a `partysplit` and `kids` a list of `partynode`s.
- `party(node, data, fitted = NULL, ...)` where `node` is a `partynode` and `data` the corresponding (learning) data (optionally without any rows) and `fitted` the corresponding fitted node ids.

Additionally: All three objects have an `info` slot where optionally arbitrary information can be stored.

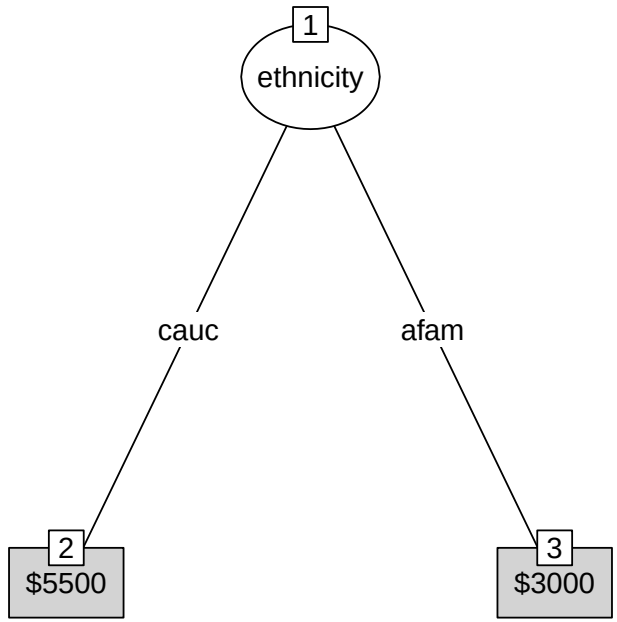
partykit: Base classes

```
> str(ms63d)
'data.frame': 0 obs. of  5 variables:
 $ earnings  : num
 $ ethnicity : Factor w/ 2 levels "cauc","afam":
 $ age       : num
 $ occupation: Factor w/ 2 levels "other","farmer":
 $ education : Ord.factor w/ 4 levels "elementary"<"highschool"<..:
```

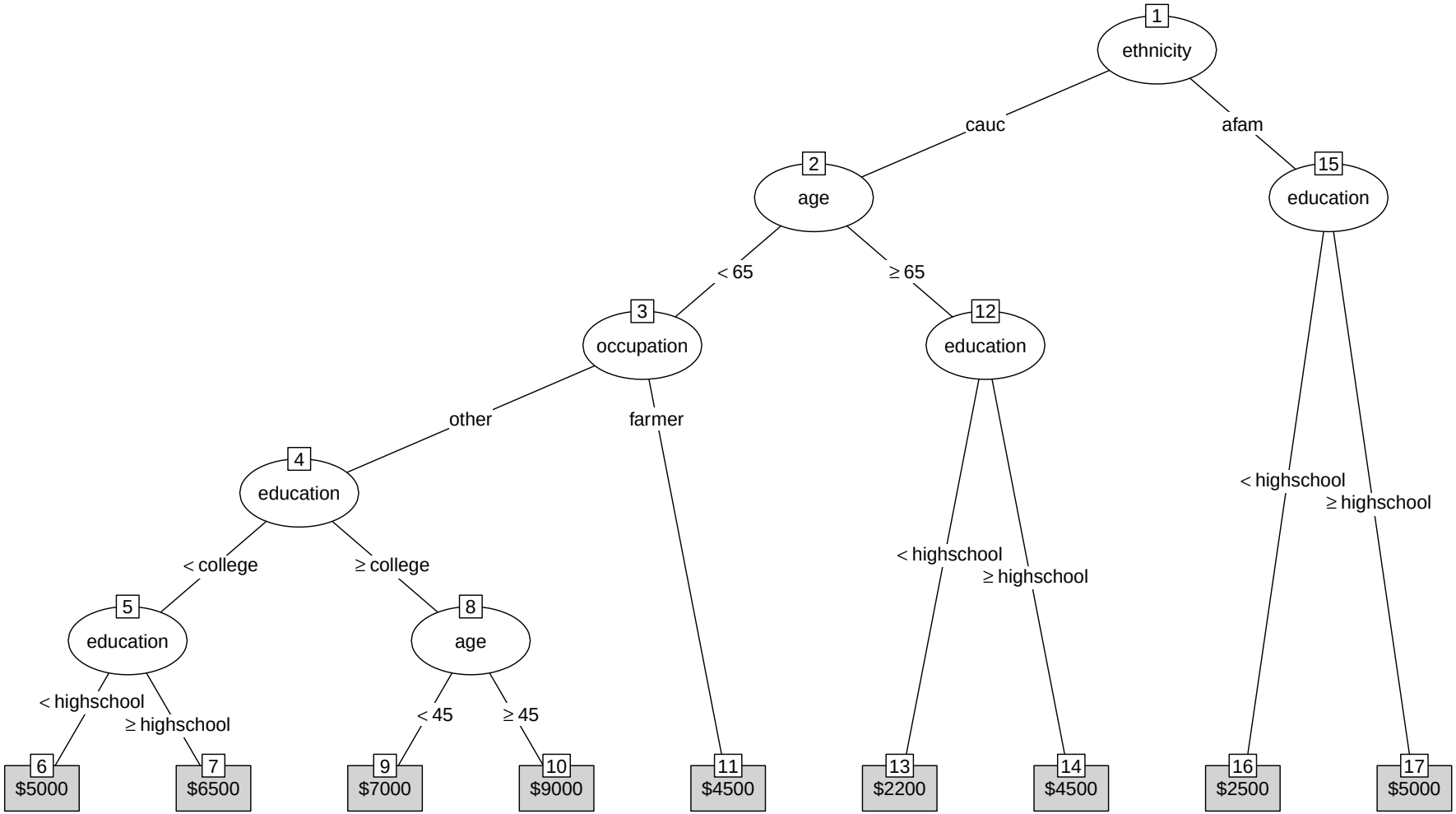
```
> pn <- partynode(1L,
+   split = partysplit(2L, index = 1:2),
+   kids = list(
+     partynode(2L, info = "$5500"),
+     partynode(3L, info = "$3000")
+   )
+ )
> py <- party(pn, ms63d)
> print(py)
[1] root
|   [2] ethnicity in cauc: $5500
|   [3] ethnicity in afam: $3000
```

```
> plot(py)
```

partykit: Base classes



partykit: Base classes



partykit: Further classes and methods

Further classes: For trees with constant fits in each terminal node, both inheriting from `party`.

- `constparty`: Stores full observed response and fitted terminal nodes in `fitted`; predictions are computed from empirical distribution of the response.
- `simpleparty`: Stores only one predicted response value along with some summary details (such as error and sample size) for each terminal node in the corresponding `info`.

Methods:

- Display: `print`, `plot`, `predict`.
- Query: `length`, `width`, `depth`, `names`, `nodeids`.
- Extract: `[`, `[[`, `nodeapply`.
- Coercion: `as.party`.

partykit: Illustration

Intention:

- Illustrate several trees using the same data.
- Here: Titanic survival data.
- In case you are not familiar with it: Survival status, gender, age (child/adult), and class (1st, 2nd, 3rd, crew) for the 2201 persons on the ill-fated maiden voyage of the Titanic.

Question: Who survived? Or how does the probability of survival vary across the covariates?

partykit: Interface to `rpart`

CART: Apply `rpart` to preprocessed `ttnc` data (see `constparty` vignette in partykit).

```
> rp <- rpart(Survived ~ Gender + Age + Class, data = ttnc)
```

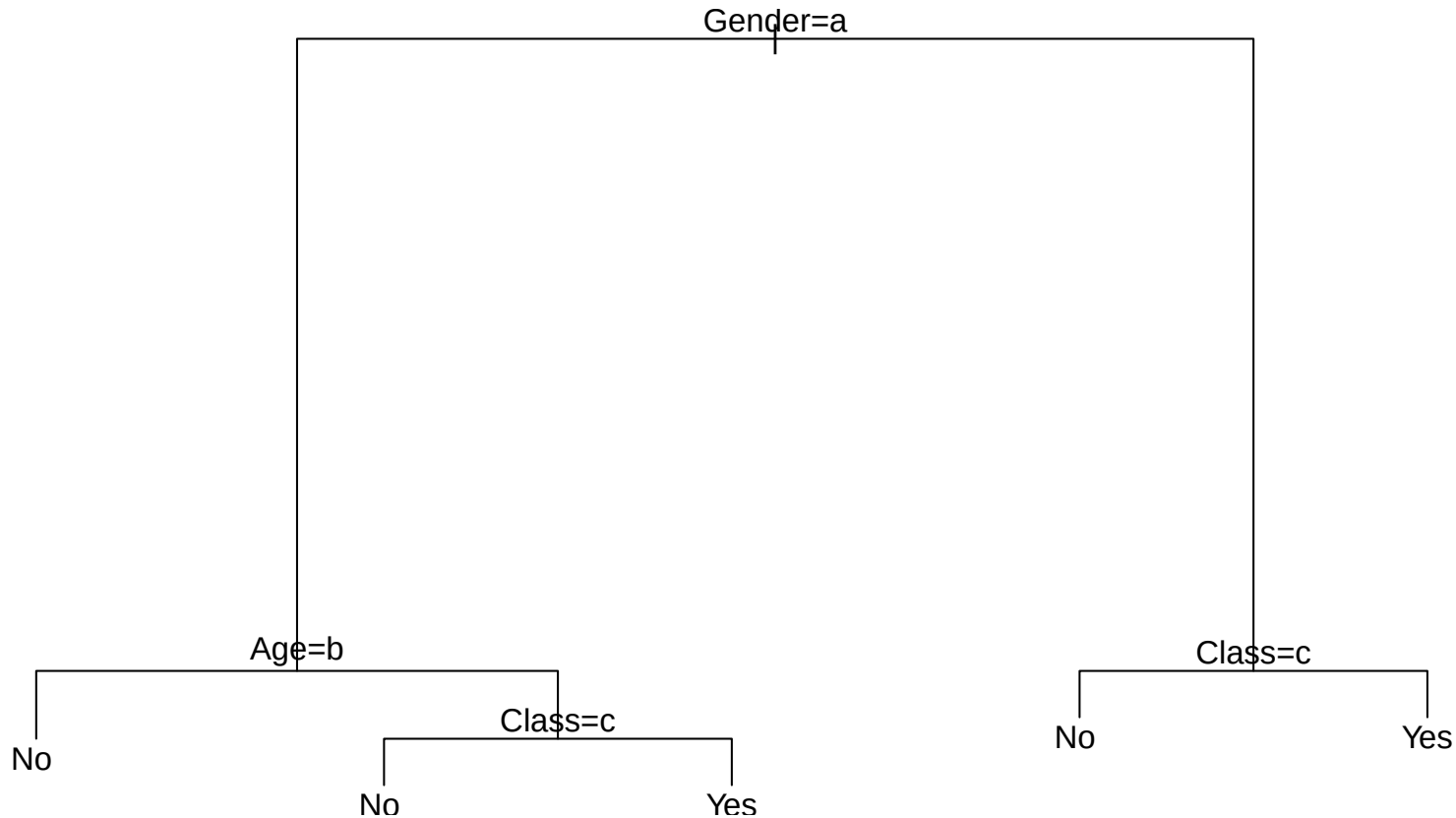
Standard plot:

```
> plot(rp)
> text(rp)
```

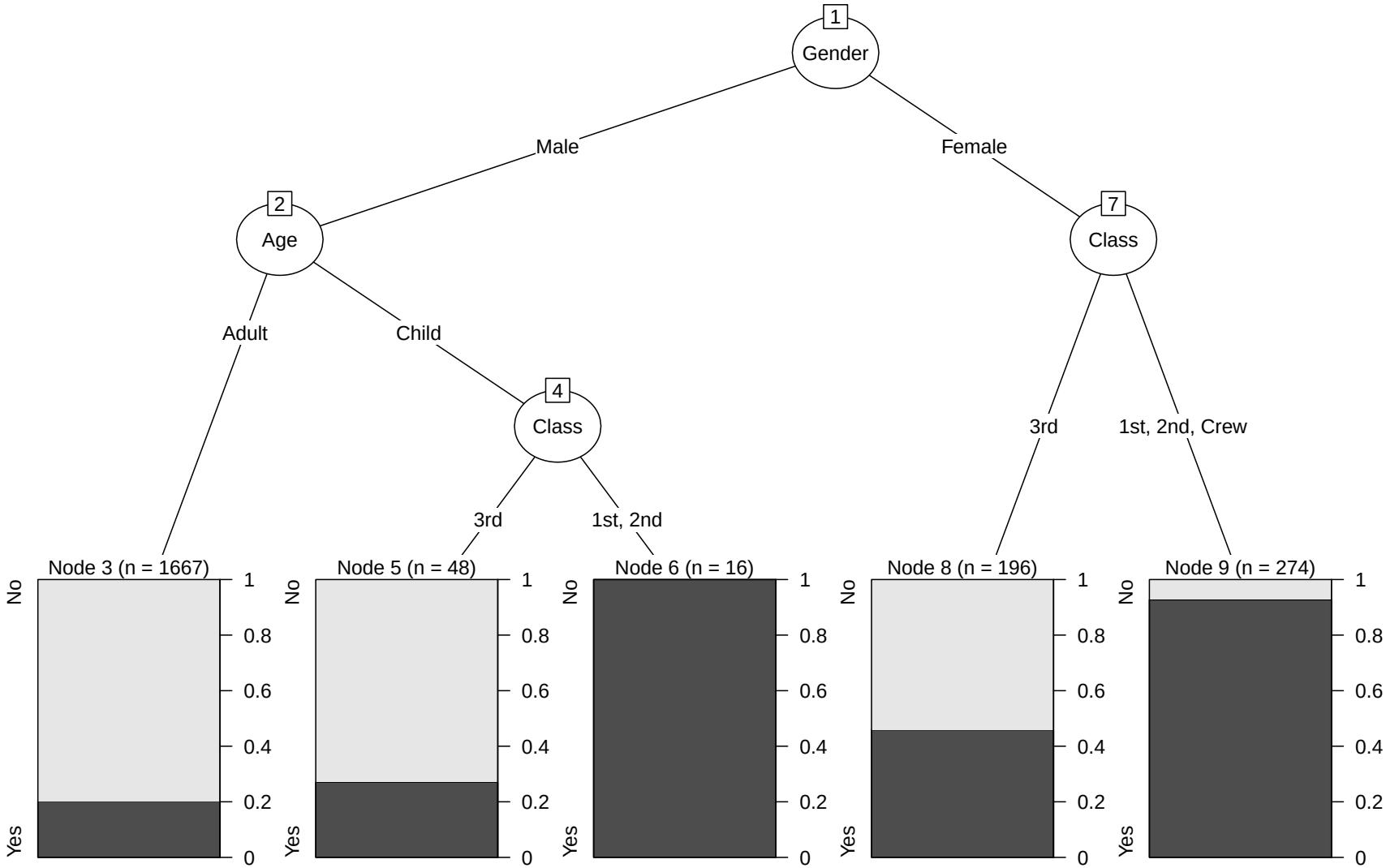
Visualization via partykit:

```
> plot(as.party(rp))
```

partykit: Interface to `rpart`



partykit: Interface to `rpart`



partykit: Interface to `rpart`

```
> rp

n= 2201

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2201 711 No (0.6769650 0.3230350)
   2) Gender=Male 1731 367 No (0.7879838 0.2120162)
     4) Age=Adult 1667 338 No (0.7972406 0.2027594) *
     5) Age=Child 64  29 No (0.5468750 0.4531250)
      10) Class=3rd 48  13 No (0.7291667 0.2708333) *
      11) Class=1st,2nd 16   0 Yes (0.0000000 1.0000000) *
   3) Gender=Female 470 126 Yes (0.2680851 0.7319149)
     6) Class=3rd 196  90 No (0.5408163 0.4591837) *
     7) Class=1st,2nd,Crew 274  20 Yes (0.0729927 0.9270073) *
```

partykit: Interface to `rpart`

```
> as.party(rp)

Model formula:
Survived ~ Gender + Age + Class

Fitted party:
[1] root
|   [2] Gender in Male
|   |   [3] Age in Adult: No (n = 1667, err = 20.3%)
|   |   [4] Age in Child
|   |   |   [5] Class in 3rd: No (n = 48, err = 27.1%)
|   |   |   [6] Class in 1st, 2nd: Yes (n = 16, err = 0.0%)
|   [7] Gender in Female
|   |   [8] Class in 3rd: No (n = 196, err = 45.9%)
|   |   [9] Class in 1st, 2nd, Crew: Yes (n = 274, err = 7.3%)

Number of inner nodes:    4
Number of terminal nodes: 5
```

partykit: Interface to `rpart`

Prediction: Compare rpart's C code and partykit's R code

```
> nd <- ttnc[rep(1:nrow(ttnc), 100), ]
> system.time(p1 <- predict(rp, newdata = nd, type = "class"))

   user  system elapsed
  0.964   0.008   0.973

> system.time(p2 <- predict(as.party(rp), newdata = nd))

   user  system elapsed
  0.184   0.000   0.184

> table(rpart = p1, party = p2)

     party
rpart     No    Yes
  No  191100      0
  Yes      0  29000
```

partykit: Interface to `J48`

J4.8: Open-source implementation of C4.5 in RWeka.

```
> j48 <- J48(Survived ~ Gender + Age + Class, data = ttnc)
```

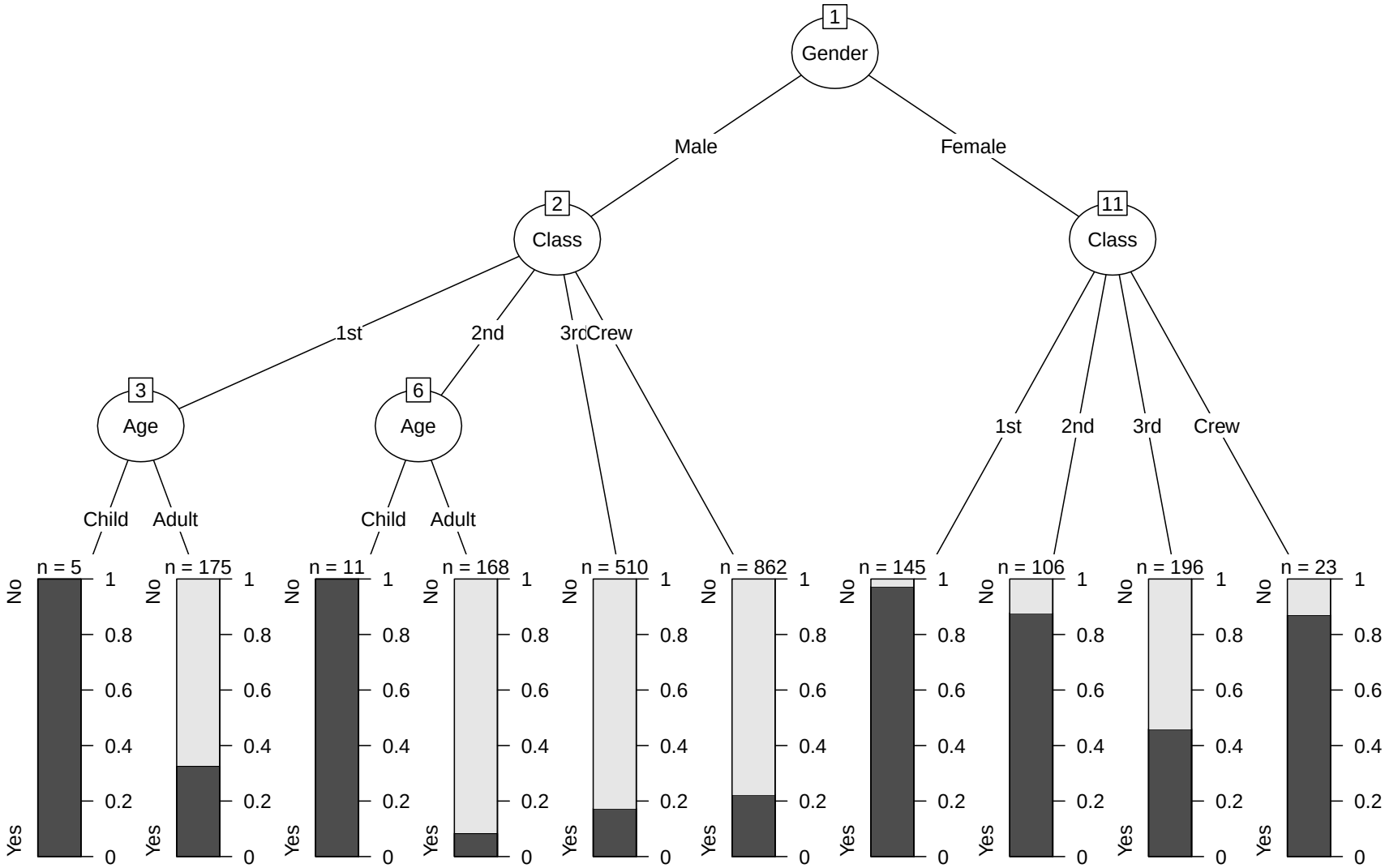
Results in a tree with multi-way splits which previously could only be displayed via Weka itself or Graphviz but not in R directly. Now:

```
> j48p <- as.party(j48)
> plot(j48p)
```

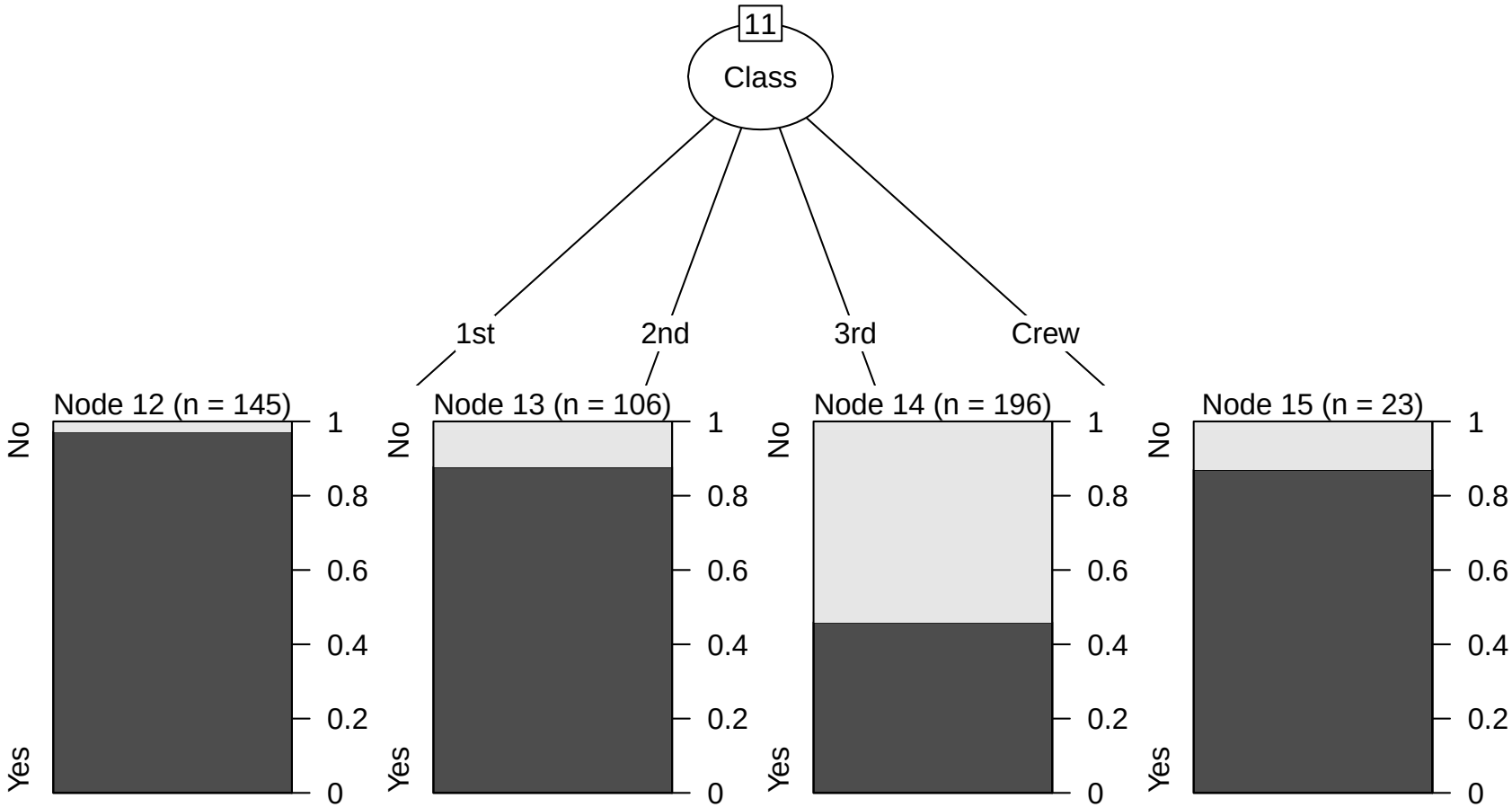
Or just a subtree:

```
> plot(j48p[11])
```

partykit: Interface to `J48`



partykit: Interface to `J48`



partykit: Further interfaces

PMML: Predictive Model Markup Language. XML-based format exported by various software packages including SAS, SPSS, R/pmml. Here, import QUEST tree from SPSS.

```
> pm <- pmmlTreeModel(system.file("pmml", "ttnc.pmml",
+   package = "partykit"))
```

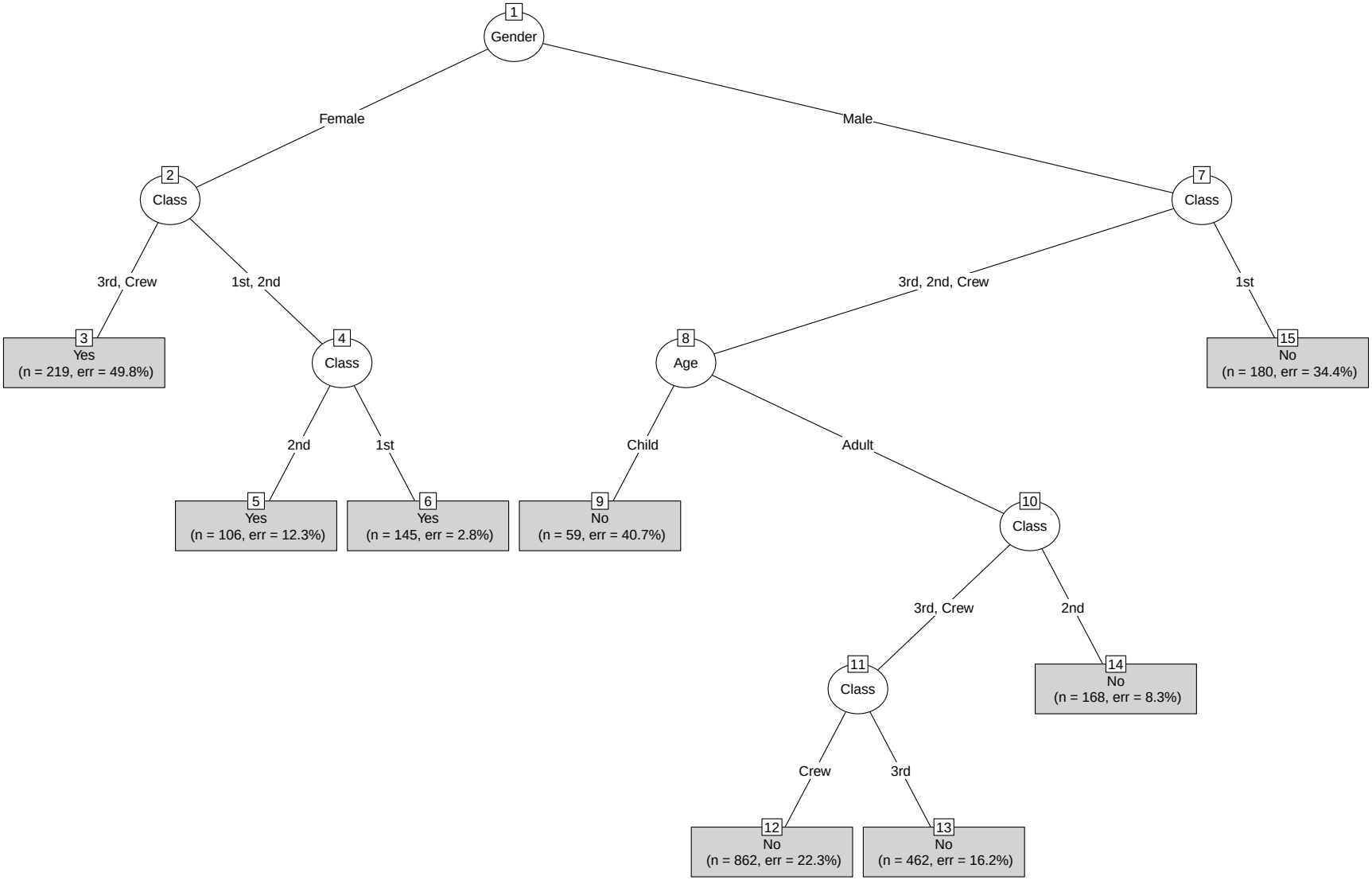
evtree: Evolutionary learning of globally optimal trees, directly using partykit.

```
> ev <- evtree(Survived ~ Gender + Age + Class,
+   data = ttnc, maxdepth = 3)
```

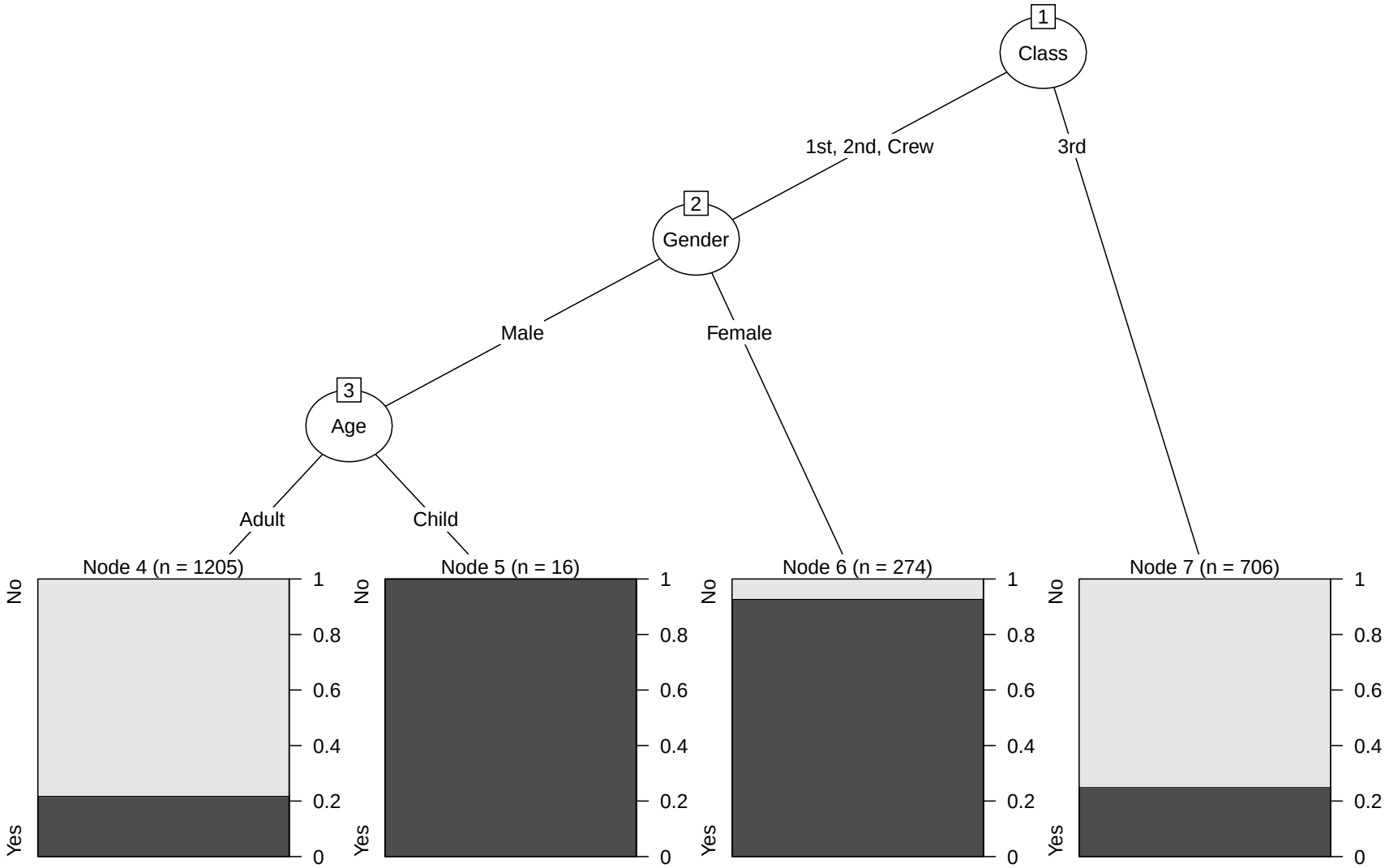
CTree: Conditional inference trees `ctree` are reimplemented more efficiently within partykit.

CHAID: R package on R-Forge, directly using partykit. (Alternatively, use SPSS and export via PMML.)

partykit: QUEST via PMML



partykit: evtree



Your first partykit tree

Finally, we want to set-up a little program that implements the following tree algorithm for the Titanic data

- In each node, do
  - For each explanatory variable, do
    - compute the minimal χ^2 p-value over all possible splits in two groups
  - Select the best variable/split combination and implement the split
- Recurse until p-value > .01 or sample size too small.

We only need three little functions:

- `findsplit`
- `growtree`
- `mytree`

findsplit

```
findsplit <- function(response, data, weights) {
  ### response: name of the response variable
  ### data: data.frame with all variables
  ### weights: case weights for current node

  ### ... some computations

  ### return split as partysplit object
  return(partysplit(
      varid = as.integer(xselect),          ### which variable?
      index = levels(data[[xselect]]) %in%
              splitpoint + 1L,              ### which split point?
      info = list(p.value = exp(logpmin))   ### save p-value
  )))
}
```

growtree

```
growtree <- function(id = 1L, response, data, weights) {
  ### recursive function of
  ### id: node identifier
  ### response: name of the response variable
  ### data: data.frame with all variables
  ### weights: case weights for current node

  ### ... some computations, call to findsplit()
  ### and growtree()

  ### return nodes
  return(partynode(id = as.integer(id), split = sp, kids = kids))
}
```

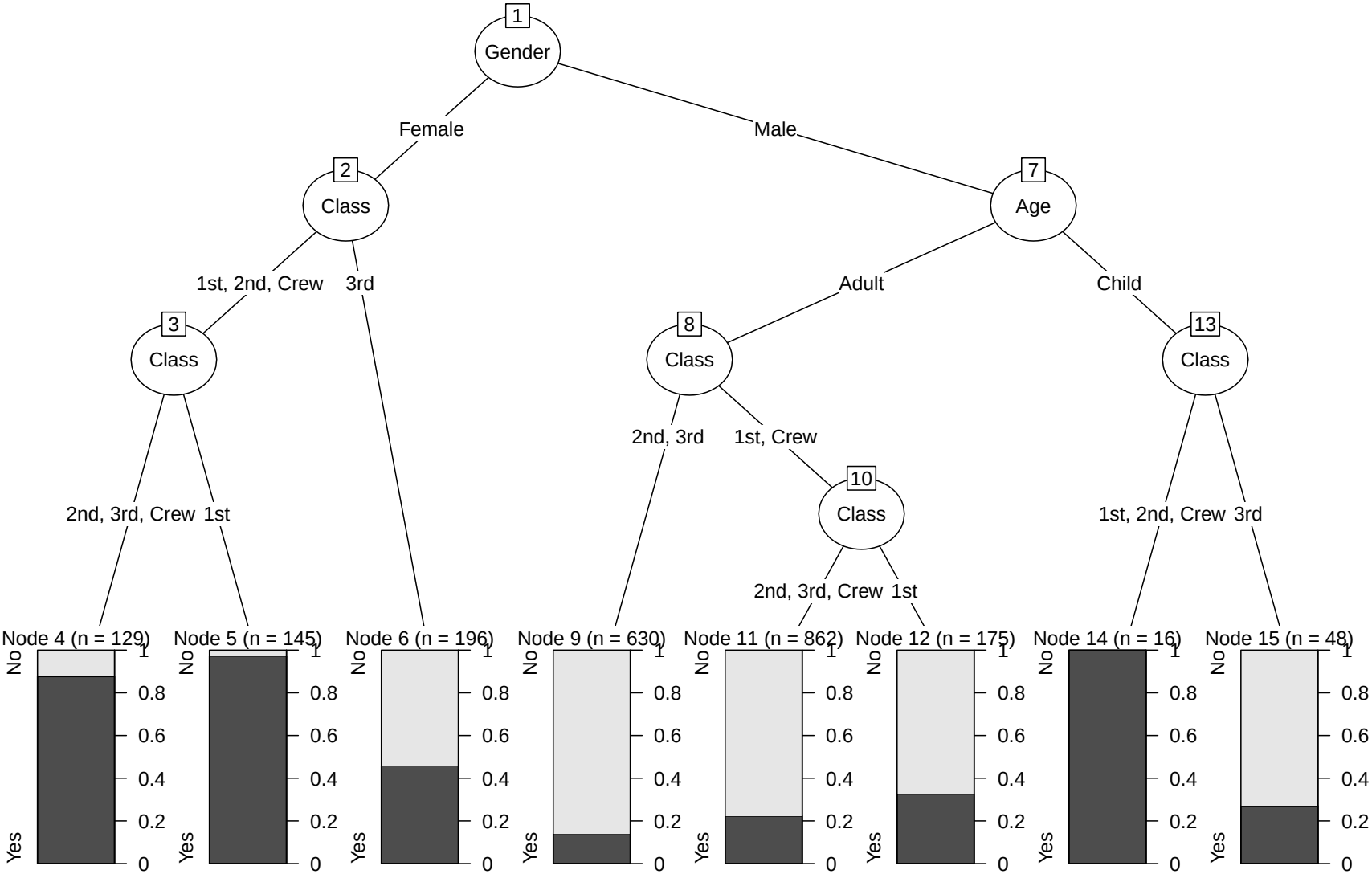
mytree

```
mytree <- function(formula, data, weights = NULL) {
  ### formula: a model formula
  ### data: data.frame with all variables
  ### weights: case weights for root node

  ### ... some computations

  nodes <- growtree(id = 1L, response, data, weights)

  ### ... some computations

  ### return rich constparty object
  ret <- party(nodes, data = data,
    fitted = data.frame(
      "(fitted)" = fitted,
      "(response)" = data[[response]],
      "(weights)" = weights,
      check.names = FALSE),
    terms = terms(formula))
  as.constparty(ret)
}
```

OK, fire away!

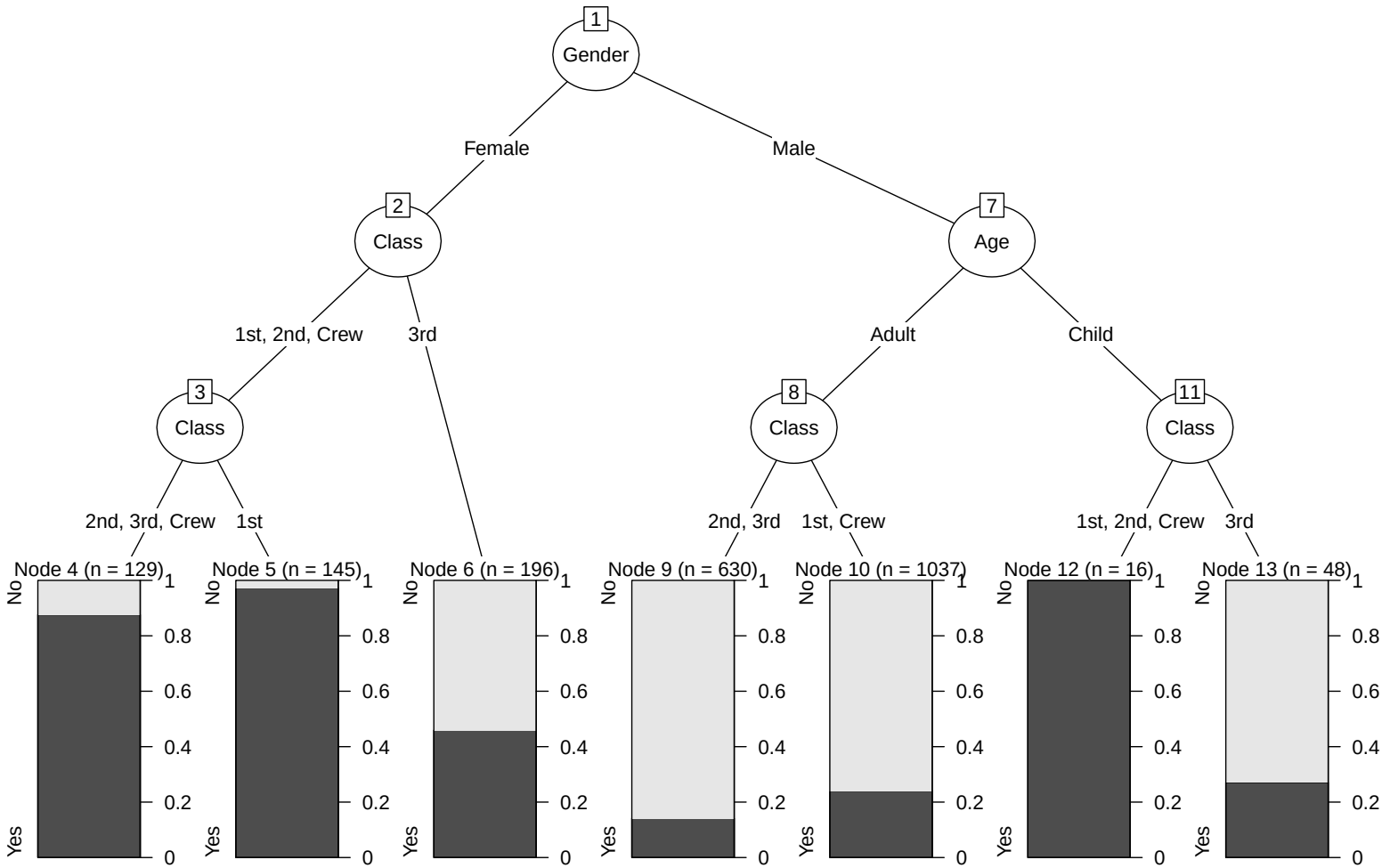
```
> plot(mytree(Survived ~ Class + Age + Gender, data = ttnc))
```

Your first partykit tree



Too big a tree? Prune!

```
> plot(nodeprune(myttnc, 10))
```



What's next?

Model-based recursive partitioning: Trees with parametric models in each node (e.g., based on least squares or maximum likelihood). Splitting based on parameter instability tests.

Talk by Achim; but I'm entitled to a coffee first...

Computational details

All examples have been produced with R 3.0.3 and packages

- partykit 0.8-0,
- rpart 4.1-6,
- RWeka 0.4-21,
- evtree 0.1-4.

All packages are freely available under the GPL from http://R-forge.R-project.org/ (partykit 0.8-0) or http://CRAN.R-project.org/.

References

Hothorn T, Zeileis A (2014). *partykit: A Toolkit for Recursive Partytioning.* R package vignette version 0.8-0.
URL https://r-forge.r-project.org/projects/partykit

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674.
10.1198/106186006X133933

Grubinger T, Zeileis A, Pfeiffer KP (2011). "evtree: Evolutionary Learning of Globally Optimal Classification and Regression Trees in R." *Working Paper 2011-20*, Working Papers in Economics and Statistics, Research Platform Empirical and Experimental Economics, Universität Innsbruck.
URL http://EconPapers.RePEc.org/RePEc:inn:wpaper:2011-20