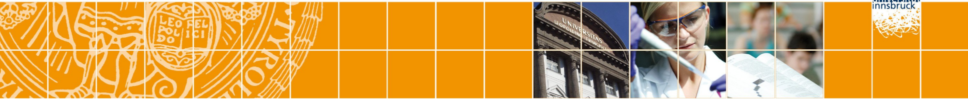
Universität Innsbruck



partykit: A Toolkit for Recursive Partytioning

Achim Zeileis, Torsten Hothorn

`http://eeecon.uibk.ac.at/~zeileis/`

Overview

- Status quo: R software for tree models
- New package: partykit
  - Unified infrastructure for recursive partytioning
  - Classes and methods
  - Interfaces to `rpart`, `J48`, ...
  - Illustrations
- Future: Next steps

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…

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

R software for tree models

Moreover: Some tree algorithms with R packages that are not on CRAN, e.g., STIMA, and TINT.

And further 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 on the learning data 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, i.e., a node can contain child nodes.
- Keep (learning) data out of the recursive node and split structure.
- Basic printing, plotting, and predicting for raw node structure.
- Customization via suitable panel or panel-generating functions.
- Coercion from existing objects (rpart, J48, etc.) 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 nodes.

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

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.
- Not use the iris data (or something from mlbench).

Solution:

- Use Titanic survival data (oh well. . .).
- 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 also below).

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

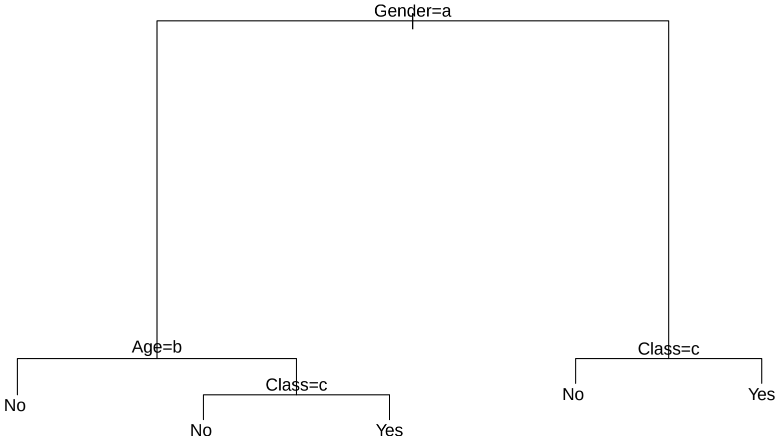
Standard plot:

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

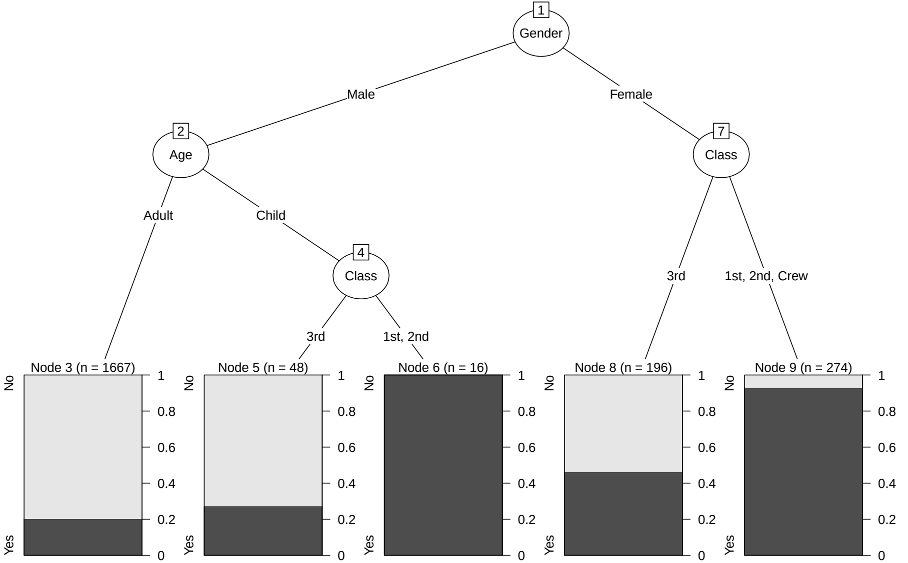
Visualization via partykit:

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

partykit: Interface to `rpart`



partykit: Interface to `rpart`



partykit: Interface to rpart

```
R> 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

```
R> 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 for (artificially) large data set.

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

   user  system elapsed
  2.816   0.020   2.835

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

   user  system elapsed
  0.456   0.000   0.454

R> 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.

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

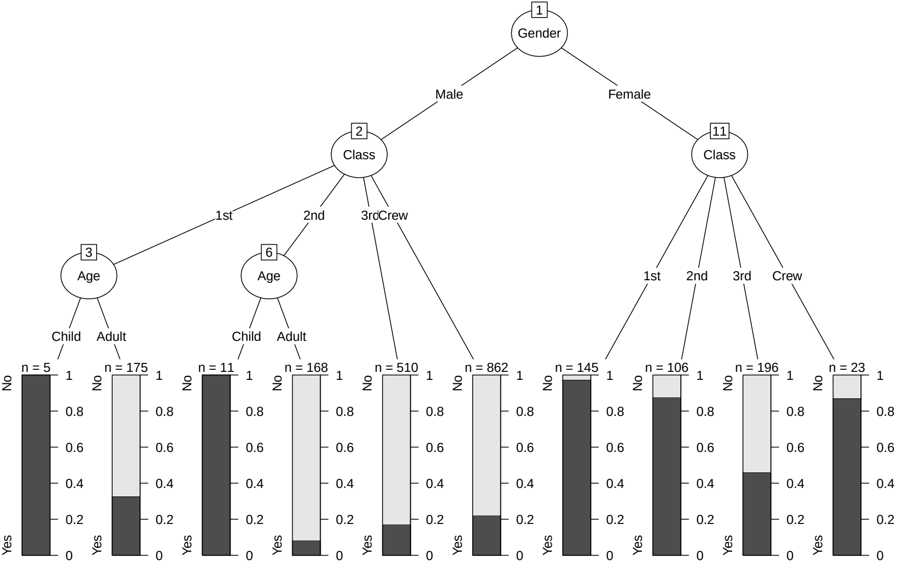
Results in a tree with multi-way splits which previously could only be displayed via Weka itself or Graphviz but not in R directly. Now:

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

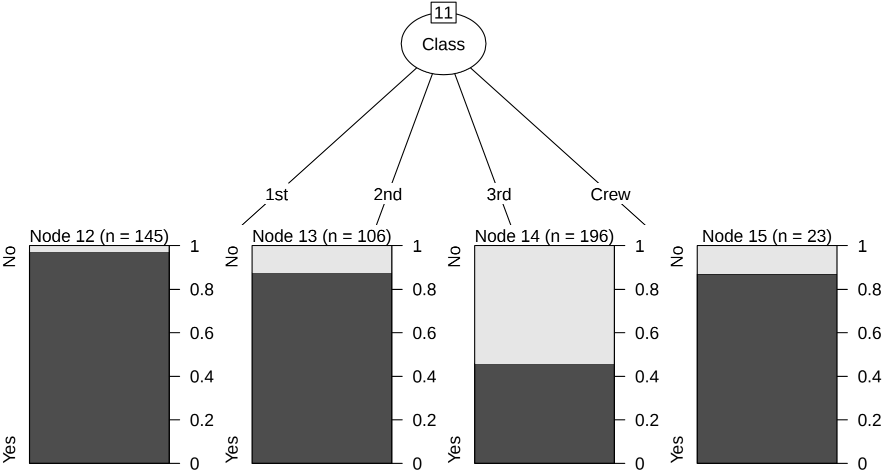
Or just a subtree:

```
R> 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, reimport CART tree.

```
R> pm <- pmmlTreeModel("ttnc.xml")
```

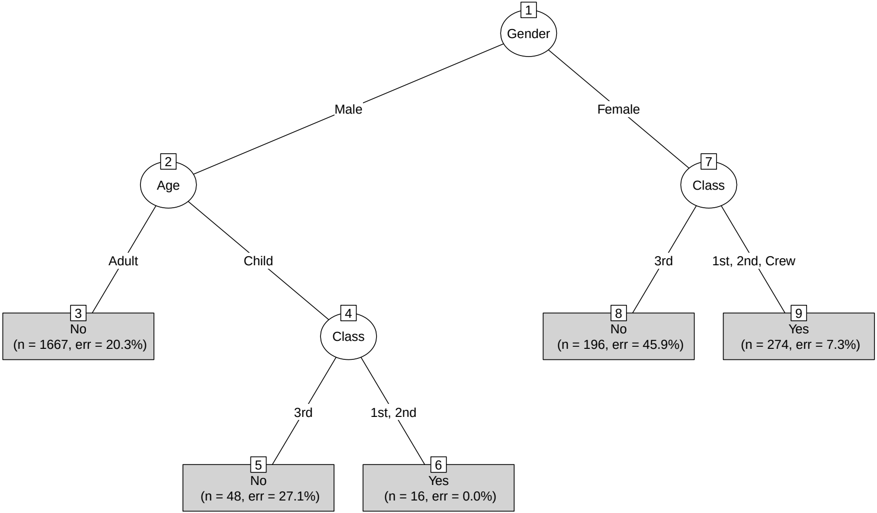
evtree: Evolutionary learning of globally optimal trees, directly using partykit.

```
R> set.seed(1071)
R> ev <- evtree(Survived ~ Gender + Age + Class, data = ttnc,
+    maxdepth = 3)
```

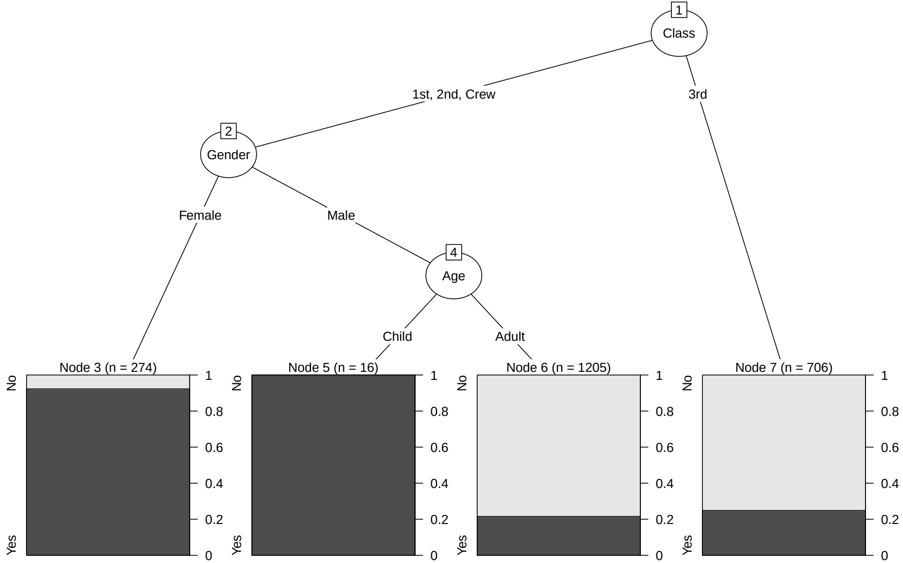
CTree: Conditional inference trees `ctree()` are reimplemented more efficiently within partykit.

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

partykit: PMML



partykit: evtree



Next steps

To do: The current code is already fairly well-tested and mostly stable. However, there are some important items on the task list.

- Extend/smooth package vignette.
- Add `xtrafo`/`ytrafo` to `ctree()` reimplementation.
- Switch `mob()` to new "`party`" class.
- Create new subclass "`modelparty`" that facilitates handling of formulas, terms, model frames, etc. for model-based trees.

Next steps

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.

Illustration: Logistic regression (or logit model), assessing differences in the effect of “preferential treatment” for women or children.

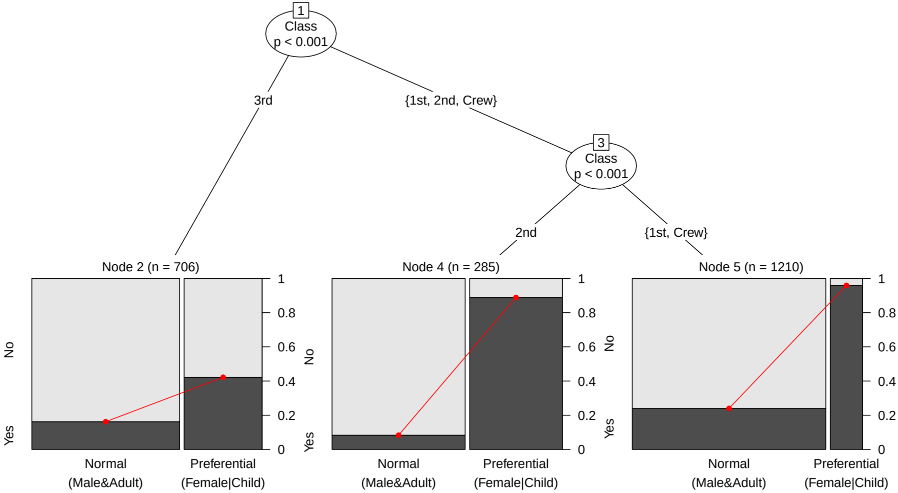
```
R> library("party")
R> mb <- mob(Survived ~ Treatment | Age + Gender + Class,
+    data = ttnc, model = glinearModel, family = binomial(),
+    control = mob_control(alpha = 0.01))
```

Odds ratio of survival given treatment differs across subsets (slope), as does the survival probability of male adults (intercept).

```
R> coef(mb)
```

```
  (Intercept) TreatmentPreferential\n(Female|Child)
2   -1.640937                              1.326906
4   -2.397895                              4.477337
5   -1.152045                              4.318123
```

Next steps


1
Class
p < 0.001
3rd
{1st, 2nd, Crew}
3
Class
p < 0.001
2nd
{1st, Crew}
Node 2 (n = 706)
Node 4 (n = 285)
Node 5 (n = 1210)
No
Yes
1
0.8
0.6
0.4
0.2
0
Normal
(Male&Adult)
Preferential
(Female|Child)

Computational details

Software: All examples have been produced with R 2.14.0 and packages partykit 0.1-2, rpart 3.1-50, RWeka 0.4-9, evtree 0.1-1, party 0.9-99995. All packages are freely available under the GPL from `http://CRAN.R-project.org/`.

Data: The contingency table `Titanic` is transformed to the data frame `ttnc` using the following code.

```
R> data("Titanic", package = "datasets")
R> ttnc <- as.data.frame(Titanic)
R> ttnc <- ttnc[rep(1:nrow(ttnc), ttnc$Freq), 1:4]
R> names(ttnc)[2] <- "Gender"
R> ttnc <- transform(ttnc, Treatment = factor(
+    Gender == "Female" | Age == "Child",
+    levels = c(FALSE, TRUE),
+    labels = c("Normal\n(Male&Adult)", "Preferential\n(Female|Child)")
+  ))
```

The last transformation also adds the treatment variable for the model-based recursive partitioning analysis.

References

Hothorn T, Zeileis A (2011). *partykit: A Toolkit for Recursive Partytioning.* R package vignette version 0.1-2. URL `http://CRAN.R-project.org/package=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. `doi: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`

Zeileis A, Hothorn T, Hornik K (2008). “Model-Based Recursive Partitioning.” *Journal of Computational and Graphical Statistics*, **17**(2), 492–514. `doi:10.1198/106186008X319331`