# partykit:

# A Toolbox for Recursive Partytioning

Torsten Hothorn
Ludwig-Maximilians-Universität München

Achim Zeileis
WU Wirtschaftsuniversität Wien

## Trees in R

The Machine Learning task view lists the following tree-related packages

- **rpart** (CART)
- **tree** (CART)
- **mvpart** (multivariate CART)
- **knnTree** (nearest-neighbor-based trees)
- **RWeka** (J4.8, M5$'$, LMT)
- **LogicReg** (Logic Regression)
- **BayesTree** (with Bayesian flavor)
- **TWIX** (with extra splits)
- **party** (conditional inference trees, model-based trees)
- ...

In addition, some packages implementing ensemble methods are based on trees as well, for example

## Trees in R

- **randomForest** (CART-based random forests)
- **randomSurvivalForest** (for censored responses)
- **party** (conditional random forests)
- **gbm** (tree-based gradient boosting)
- **mboost** (model-based and tree-based gradient boosting)
- ...

These packages deal with very similar problems. However, similar or even the same tasks are implemented differently by each package. No code reuse takes place.

## Missing in Action

Moreover, important methods in this field, mostly from the statistics literature, are missing from the above list, for example

- CHAID
- FACT
- QUEST
- GUIDE
- LOTUS
- CRUISE
- ...

## Adding more Trees?

It is hard to come up with implementations of new tree-based algorithms, because one needs to take care not only of fitting the tree but also of

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

## Wouldn't it be nice to...

... have a common infrastructure for

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

## R-Forge Package partykit

offers a unified approach to

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

## Unified Representation of Trees

The most important classes in **partykit** are

`partysplit`: binary, multiway and functional splits in ordered and unordered variables,

`partynode`: inner nodes containing splits, surrogate splits, kid nodes, and terminal nodes containing models or constant fits,

`party`: trees (nodes and data).

The API essentially consists of the corresponding constructors.

## Methods

Coercing:

`as.party.rpart`, `as.party.J48`, `as.party.pmmlTreeModel`

Inspecting:

`print.party`, `plot.party`, both are extensible

Computing:

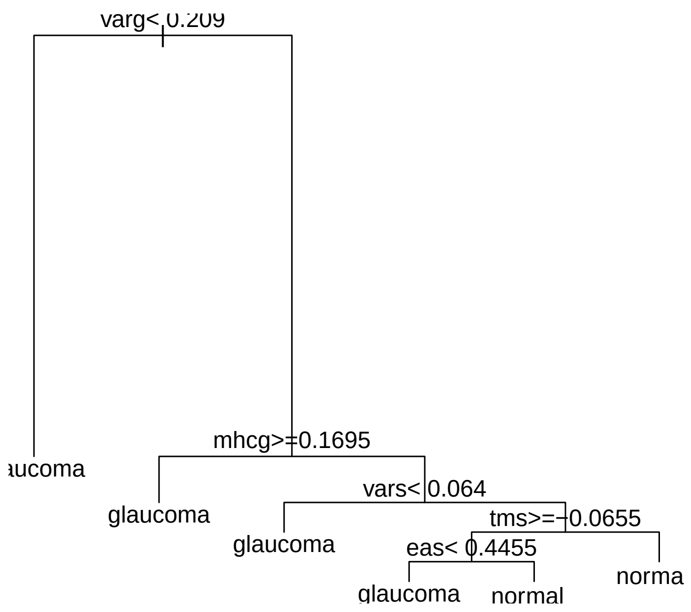
`predict.party`, subset methods

## Example: Pretty rpart Trees

```
R> data("GlaucomaM", package = "ipred")
R> library("rpart")
R> g_rpart <- rpart(Class ~ ., data = GlaucomaM)
R> print(g_rpart)
n= 196

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

 1) root 196 98 glaucoma (0.50000000 0.50000000)
   2) varg< 0.209 76  6 glaucoma (0.92105263 0.07894737) *
   3) varg>=0.209 120 28 normal (0.23333333 0.76666667)
     6) mhcg>=0.1695 7  0 glaucoma (1.00000000 0.00000000) *
     7) mhcg< 0.1695 113 21 normal (0.18584071 0.81415929)
      14) vars< 0.064 19  9 glaucoma (0.52631579 0.47368421) *
      15) vars>=0.064 94 11 normal (0.11702128 0.88297872)
        30) tms>=-0.0655 32  8 normal (0.25000000 0.75000000)
          60) eas< 0.4455 7  2 glaucoma (0.71428571 0.28571429) *
          61) eas>=0.4455 25  3 normal (0.12000000 0.88000000) *
        31) tms< -0.0655 62  3 normal (0.04838710 0.95161290) *
```

## Example: Pretty rpart Trees

```
R> plot(g_rpart)
R> text(g_rpart)
```

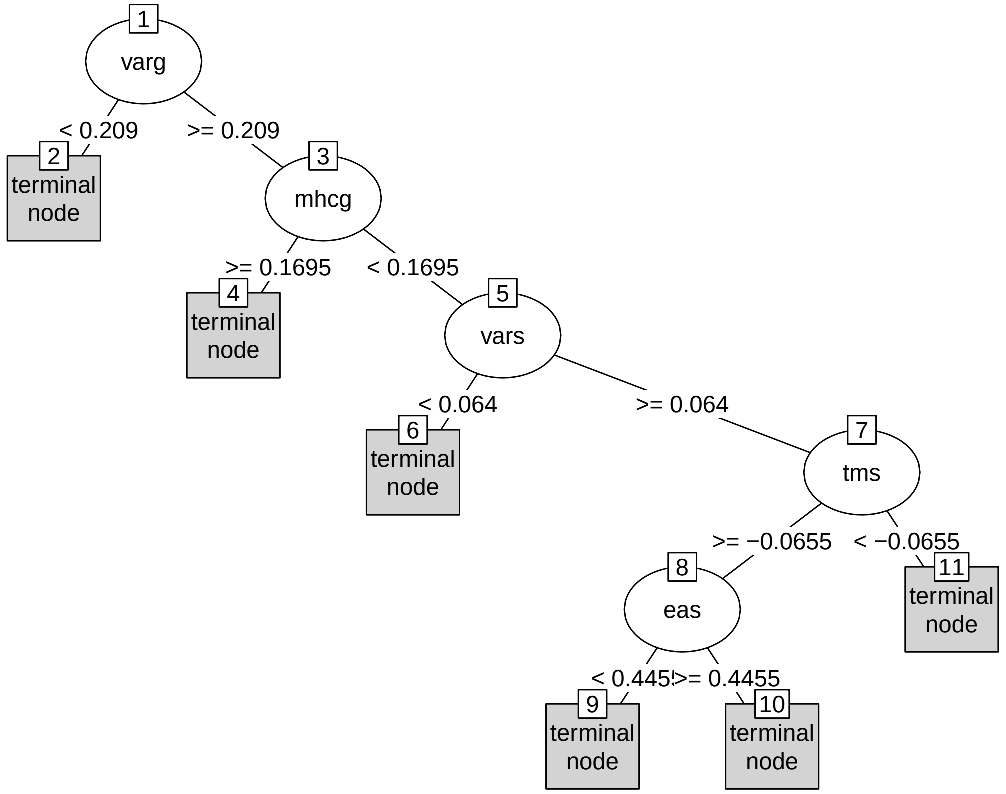


## Example: Pretty rpart Trees

```
R> library("partykit")
R> g_party <- as.party(g_rpart)
R> print(g_party, header = FALSE)
[1] root
|   [2] varg < 0.209: glaucoma (n = 76, err = 7.9%)
|   [3] varg >= 0.209
|   |   [4] mhcg >= 0.1695: glaucoma (n = 7, err = 0.0%)
|   |   [5] mhcg < 0.1695
|   |   |   [6] vars < 0.064: glaucoma (n = 19, err = 47.4%)
|   |   |   [7] vars >= 0.064
|   |   |   |   [8] tms >= -0.0655
|   |   |   |   |   [9] eas < 0.4455: glaucoma (n = 7, err = 28.6%)
|   |   |   |   |   [10] eas >= 0.4455: normal (n = 25, err = 12.0%)
|   |   |   |   [11] tms < -0.0655: normal (n = 62, err = 4.8%)

Number of inner nodes:    5
Number of terminal nodes: 6
```

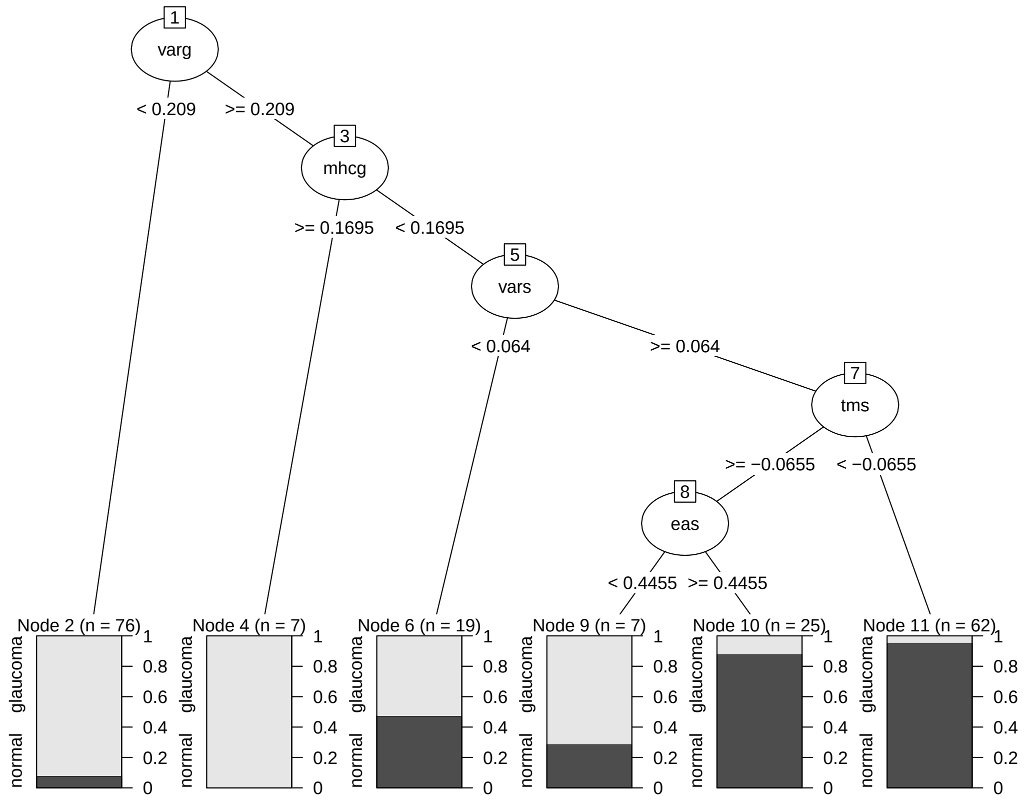
## Example: Pretty rpart Trees

```
R> plot(g_party, type = "simple")
```



## Example: Pretty rpart Trees

```
R> plot(g_party, type = "extended")
```



## Example: Fast Predictions

```
R> g_J48 <- J48(Class ~ ., data = GlaucomaM)
R> g_party <- as.party(g_J48)
R> nd <- GlaucomaM[rep(1:nrow(GlaucomaM), 100),]
R> system.time(p1 <- predict(g_J48, newdata = nd))
   user  system elapsed
  1.468   0.156   1.623
R> system.time(p2 <- predict(g_party, newdata = nd))
   user  system elapsed
  0.096   0.000   0.097
R> table(p1, p2)
          p2
p1          glaucoma normal
  glaucoma      9900      0
  normal           0   9700
```

## Example: CHAID

**partykit** makes it ‘easy’ to implement new or old tree algorithms.

My students implemented CHAID as a software project last winter.
I wanted them to focus on the algorithm, not on technical details.

They only had to implement variable selection, splitting and stopping rules, **partykit** takes care of the rest.

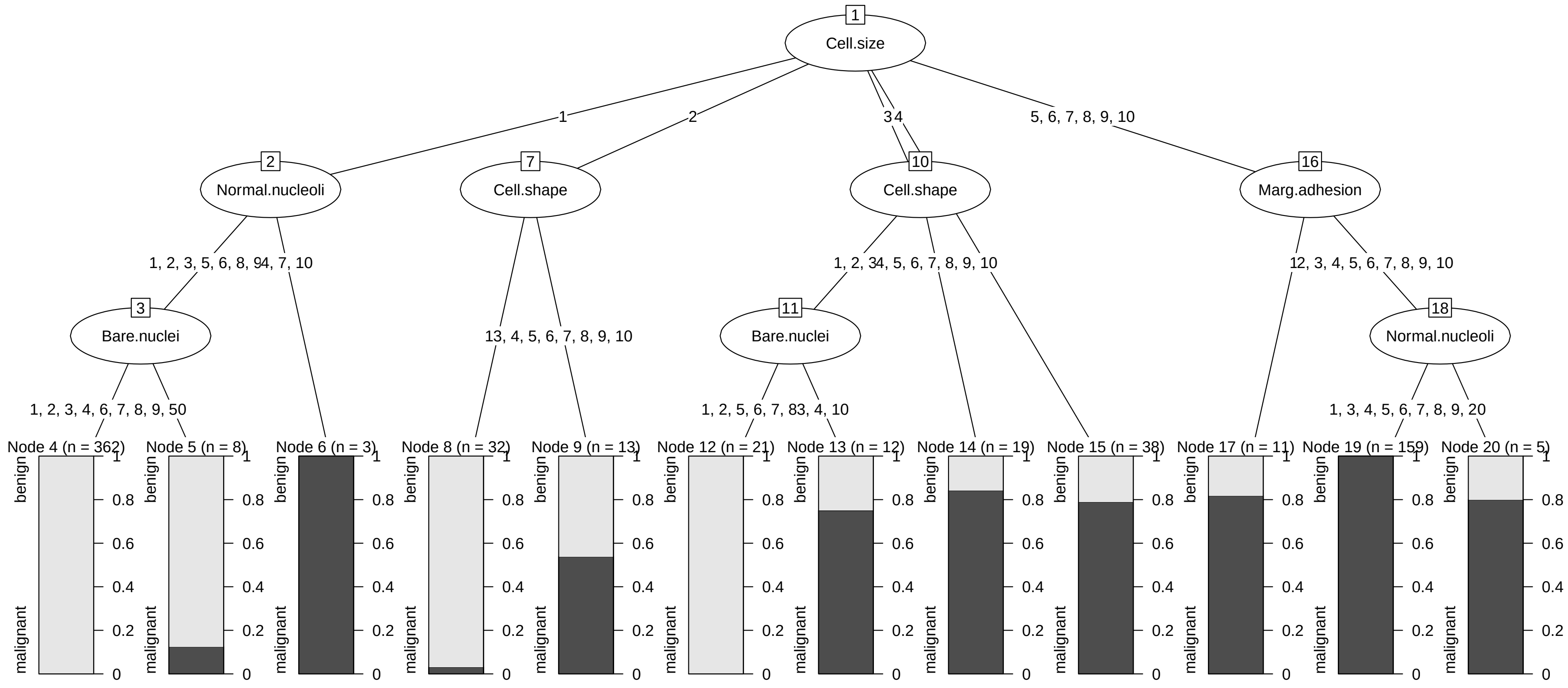
The resulting package **CHAID** is available from R-Forge and consists of only 362 lines of R code and was validated against SPSS 15.

## Example: CHAID

Fit a CHAID tree to BreastCancer data from package **mlbench**:

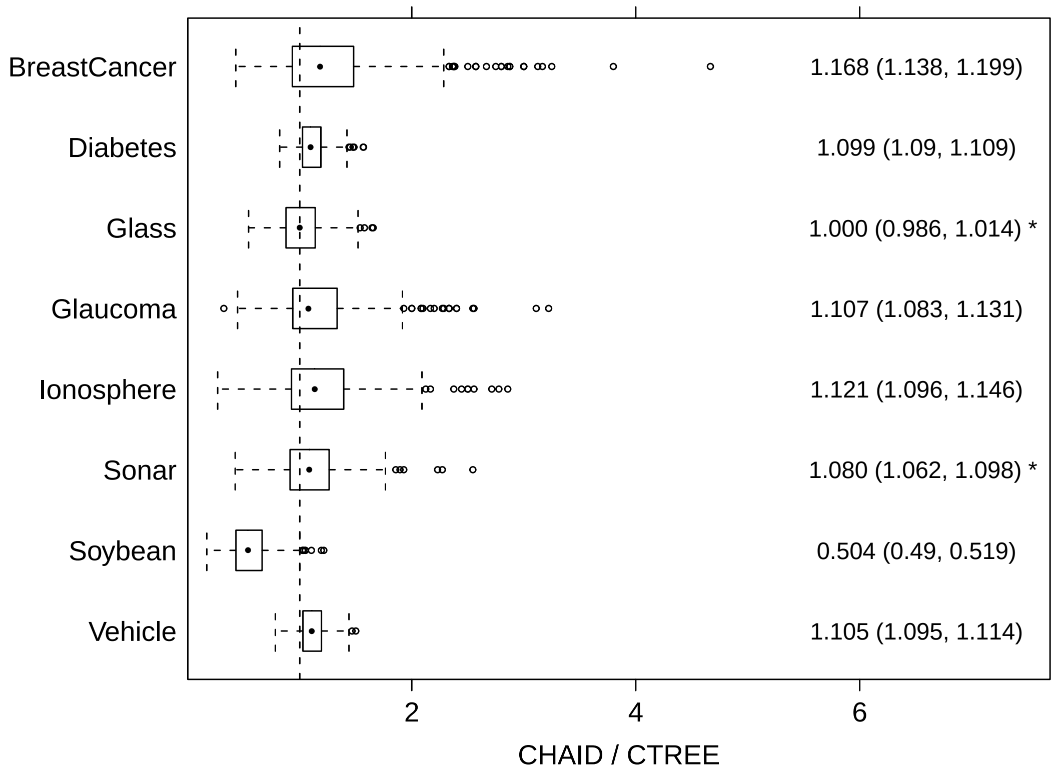
```
R> data("BreastCancer", package = "mlbench")
R> library("CHAID")
R> b_chaid <- chaid(Class ~ Cl.thickness + Cell.size + Cell.shape +
+                   Marg.adhesion + Epith.c.size + Bare.nuclei +
+                   Bl.cromatin + Normal.nucleoli + Mitoses,
+                   data = BreastCancer)

R> plot(b_chaid)
```

## Example: CHAID



# Example: CHAID

Benchmark experiment misclassification error CHAID vs. CTREE
(Bachelor thesis by Stefanie Thiemichen)



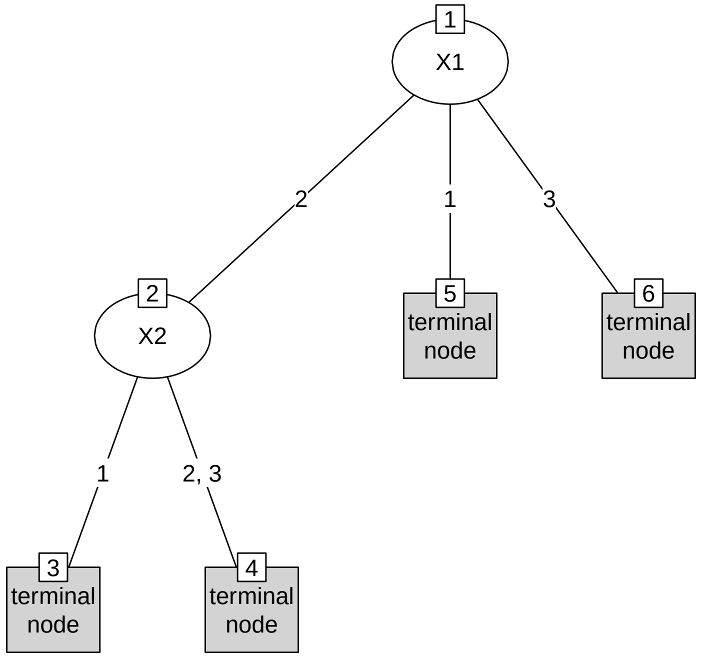
## Example: PMML & SPSS

Toy PMML example:

```
<PMML
        version="3.1"
        xmlns="http://www.dmg.org/PMML-3_1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.dmg.org/PMML-3_1 pmml-3-1.xsd">
        <Header
                copyright="Copyright (c) 1989-2006 by SPSS Inc. All rights reserved.">
                <Application
                        name="SPSS for Microsoft Windows"
                        version="15.0.1"/>
                <Timestamp/>
        </Header>
        <DataDictionary
                numberOfFields="4">
                <DataField
...
```

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

## Example: PMML & SPSS



## Example: PMML & SPSS

```
R> nd <- data.frame(X1 = gl(3, 10), X2 = sample(gl(3, 10)))
R> predict(pm, newdata = nd)
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
27 28 29 30
 1  1  1  1
Levels: 1 2
```

## What's next?

Once a stable version of **partykit** is available from CRAN, we hope that useRs will start

- adding new tree algorithms,
- adding new ensemble methods,
- adding coercing methods from/to other packages,
- adding new plot functionality, and generally
- will use our kit to party harder.