Parties, Models, Mobsters
A New Implementation of Model-Based Recursive Partitioning in R

Achim Zeileis, Torsten Hothorn

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

- Model-based recursive partitioning
  - A generic approach
  - Example: Bradley-Terry trees

- Implementation in R
  - Building blocks: Parties, models, mobsters
  - Old implementation in party
  - All new implementation in partykit

- Illustration
Model-based recursive partitioning

**Models:** Estimation of parametric models with observations $y_i$ (and regressors $x_i$), parameter vector $\theta$, and additive objective function $\Psi$.

$$\hat{\theta} = \arg\min_{\theta} \sum_i \Psi(y_i, x_i, \theta).$$

**Recursive partitioning:**

1. Fit the model in the current subsample.
2. Assess the stability of $\theta$ across each partitioning variable $z_j$.
3. Split sample along the $z_{j*}$ with strongest association: Choose breakpoint with highest improvement of the model fit.
4. Repeat steps 1–3 recursively in the subsamples until some stopping criterion is met.
Model-based recursive partitioning

Parameter instability tests:

- Based on empirical estimating functions (or score/gradient contributions): $\Psi'(y_i, x_i, \hat{\theta})$.
- Under parameter stability: $\Psi'$ fluctuates randomly around its expectation zero.
- Under parameter instability: Systematic departures from zero in subsamples.
- Hence fluctuation can be captured across numeric partitioning variables or within levels of categorical partitioning variables.
- Bonferroni correction for testing across multiple partitioning variables.
Questions: Which of these women is more attractive? How does the answer depend on age, gender, and the familiarity with the associated TV show Germany’s Next Topmodel?
Bradley-Terry trees

Node 1 (age $p < 0.001$)
  - $\leq 52$
  - $> 52$

Node 2 (age $p = 0.017$)
  - $q2$

Node 3 (n = 35)
- Male
- Female

Node 4 (gender $p = 0.007$)
- Male
- Female

Node 5 (n = 71)

Node 6 (n = 56)

Node 7 (n = 30)
Implementation: Building blocks

**Workhorse function:** `mob()` for
- data handling,
- calling model fitters,
- carrying out parameter instability tests and
- recursive partitioning algorithm.

**Required functionality:**
- *Models*: Fitting functions for statistical models (optimizing suitable objective function).
- *Mobsters*: High-level interfaces (`lmtree()`, `bttree()`, ...) that call lower-level `mob()` with suitable options and methods.
Implementation: Old \texttt{mob()} in \textit{party}

\textbf{Parties:} S4 class ‘BinaryTree’.
- Originally developed only for \texttt{ctree()} and somewhat “abused”.
- Rather rigid and hard to extend.

\textbf{Models:} S4 ‘StatModel’ objects.
- Intended to conceptualize unfitted model objects.
- Required some “glue code” to accomodate non-standard interface for data handling and model fitting.

\textbf{Mobsters:}
- \texttt{mob()} already geared towards (generalized) linear models.
- Other interfaces in \textit{psychotree} and \textit{betareg}.
- Hard to do fine control due to adopted S4 classes: Many unnecessary computations and copies of data.
Implementation: **New mob() in partykit**

**Parties:** S3 class ‘modelparty’ built on ‘party’.
- Separates data and tree structure.
- Inherits generic infrastructure for printing, predicting, plotting, . . .

**Models:** Plain functions with input/output convention.
- Basic and extended interface for rapid prototyping and for speeding up computings, respectively.
- Only minimal glue code required if models are well-designed.

**Mobsters:**
- mob() completely agnostic regarding models employed.
- Separate interfaces lmtree(), glmtree(), . . .
- New interfaces typically need to bring their model fitter and adapt the main methods print(), plot(), predict() etc.
Implementation: New \texttt{mob()} in \textit{partykit}

New inference options: Not used by default by optionally available.

- New parameter instability tests for ordinal partitioning variables. Alternative to unordered $\chi^2$ test but computationally intensive.
- Post-pruning based on information criteria (e.g., AIC or BIC), especially for very large datasets where traditional significance levels are not useful.
- Multiway splits for categorical partitioning variables.
- Treat weights as proportionality weights and not as case weights.
Implementation: Models

**Input:** Basic interface.

\[
\text{fit}(y, x = \text{NULL}, \text{start} = \text{NULL}, \text{weights} = \text{NULL}, \text{offset} = \text{NULL}, \ldots)
\]

y, x, weights, offset are (the subset of) the preprocessed data. Starting values and further fitting arguments are in start and \ldots.

**Output:** Fitted model object of class with suitable methods.

- \texttt{coef()}: Estimated parameters \(\hat{\theta}\).
- \texttt{logLik()}: Maximized log-likelihood function \(-\sum_i \Psi(y_i, x, \hat{\theta})\).
- \texttt{estfun()}: Empirical estimating functions \(\Psi'(y_i, x_i, \hat{\theta})\).
Implementation: Models

**Input:** Extended interface.

\[
\text{fit}(y, x = \text{NULL}, \text{start} = \text{NULL}, \text{weights} = \text{NULL}, \\
\text{offset} = \text{NULL}, \ldots, \text{estfun} = \text{FALSE}, \text{object} = \text{FALSE})
\]

**Output:** List.

- **coefficients:** Estimated parameters \( \hat{\theta} \).
- **objfun:** Minimized objective function \( \sum_i \Psi(y_i, x, \hat{\theta}) \).
- **estfun:** Empirical estimating functions \( \Psi'(y_i, x_i, \hat{\theta}) \). Only needed if \( \text{estfun} = \text{TRUE} \), otherwise optionally NULL.
- **object:** A model object for which further methods could be available (e.g., predict(), or fitted(), etc.). Only needed if \( \text{object} = \text{TRUE} \), otherwise optionally NULL.

**Internally:** Extended interface constructed from basic interface if supplied. Efficiency can be gained through extended approach.
Illustration: Bradley-Terry trees

Data, packages, and estfun() method:

```r
R> data("Topmodel2007", package = "psychotree")
R> library("partykit")
R> library("psychotools")
R> estfun.btReg <- function(x, ...) x$estfun
```

Basic model fitting function:

```r
R> btfit1 <- function(y, x = NULL, start = NULL, weights = NULL,
+ offset = NULL, ...) btReg.fit(y, weights = weights, ...)
```

Fit Bradley-Terry tree:

```r
R> system.time(bt1 <- mob(
+ preference ~ 1 | gender + age + q1 + q2 + q3,
+ data = Topmodel2007, fit = btfit1))
    user  system elapsed
 5.112   0.020   5.263
```
Illustration: Bradley-Terry trees

Extended model fitting function:

```R
R> btfit2 <- function(y, x = NULL, start = NULL, weights = NULL,
+                  offset = NULL, ..., estfun = FALSE, object = FALSE) {
+    rval <- btReg.fit(y, weights = weights, ...,
+                     estfun = estfun, vcov = object)
+    list(
+        coefficients = rval$coefficients,
+        objfun = -rval$loglik,
+        estfun = if(estfun) rval$estfun else NULL,
+        object = if(object) rval else NULL
+    )
+ }
```

Fit Bradley-Terry tree again:

```R
R> system.time(bt2 <- mob(
+    preference ~ 1 | gender + age + q1 + q2 + q3,
+    data = Topmodel2007, fit = btfit2))
```

```
user  system elapsed
4.004  0.012   4.087
```
Illustration: Bradley-Terrry trees

Model-based recursive partitioning (btfit2)

Model formula:
preference ~ 1 | gender + age + q1 + q2 + q3

Fitted party:
[1] root
| [2] age <= 52
| | [3] q2 in yes: n = 35
| | Barbara Anni Hana Fiona Mandy
| | 1.3378 1.2318 2.0499 0.8339 0.6217
| | [4] q2 in no
| | | [5] gender in male: n = 71
| | | Barbara Anni Hana Fiona Mandy
| | | 0.43866 0.03877 0.84629 0.69424 -0.10003
| | | Barbara Anni Hana Fiona Mandy
| | | 0.9475 0.7246 0.4452 0.6350 -0.4965
| [7] age > 52: n = 30
| Barbara Anni Hana Fiona Mandy
| 0.2178 -1.3166 -0.3059 -0.2591 -0.2357
Illustration: Bradley-Terry trees

Number of inner nodes: 3
Number of terminal nodes: 4
Number of parameters per node: 5
Objective function: 1829

Standard methods readily available:

R> plot(bt2)
R> coef(bt2)

<table>
<thead>
<tr>
<th></th>
<th>Barbara</th>
<th>Anni</th>
<th>Hana</th>
<th>Fiona</th>
<th>Mandy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1.3378</td>
<td>1.23183</td>
<td>2.0499</td>
<td>0.8339</td>
<td>0.6217</td>
</tr>
<tr>
<td>5</td>
<td>0.4387</td>
<td>0.08877</td>
<td>0.8463</td>
<td>0.6942</td>
<td>-0.1000</td>
</tr>
<tr>
<td>6</td>
<td>0.9475</td>
<td>0.72459</td>
<td>0.4452</td>
<td>0.6350</td>
<td>-0.4965</td>
</tr>
<tr>
<td>7</td>
<td>0.2178</td>
<td>-1.31663</td>
<td>-0.3059</td>
<td>-0.2591</td>
<td>-0.2357</td>
</tr>
</tbody>
</table>

Customization:

R> worthf <- function(info) paste(info$object$labels, +    format(round(worth(info$object), digits = 2)), sep = "": ")
R> plot(bt2, FUN = worthf)
Illustration: Bradley-Terry trees

1. \( \text{age} \)
   \( p < 0.001 \)
   \( \leq 52 \)
   \( > 52 \)

2. \( q^2 \)
   \( p = 0.017 \)

3. yes
   \( n = 35 \)
   Estimated parameters:
   Barbara 1.3378
   Anni 1.2318
   Hana 2.0499
   Fiona 0.8339
   Mandy 0.6217

4. gender
   \( p = 0.007 \)
   male
   female

5. \( n = 71 \)
   Estimated parameters:
   Barbara 0.43866
   Anni 0.08877
   Hana 0.84629
   Fiona 0.69424
   Mandy −0.10003

6. \( n = 56 \)
   Estimated parameters:
   Barbara 0.9475
   Anni 0.7246
   Hana 0.4452
   Fiona 0.6350
   Mandy −0.4965

7. \( n = 30 \)
   Estimated parameters:
   Barbara 0.2178
   Anni −1.3166
   Hana −0.3059
   Fiona −0.2591
   Mandy −0.2357
Illustration: Bradley-Terry trees

3

Objects

Worth parameters

0.0 0.1 0.2 0.3 0.4

●

●

●

●

●

Barbara Anni Hana Fiona Mandy Anja

5

Objects

Worth parameters

0.0 0.1 0.2 0.3 0.4

●

●

●

●

●

Barbara Anni Hana Fiona Mandy Anja

6

Objects

Worth parameters

0.0 0.1 0.2 0.3 0.4

●

●

●

●

●

Barbara Anni Hana Fiona Mandy Anja

7

Objects

Worth parameters

0.0 0.1 0.2 0.3 0.4

●

●

●

●

●

Barbara Anni Hana Fiona Mandy Anja
Illustration: Bradley-Terry trees

Apply plotting in all terminal nodes:

```r
R> par(mfrow = c(2, 2))
R> nodeapply(bt2, ids = c(3, 5, 6, 7), FUN = function(n) + plot(n$info$object, main = n$id, ylim = c(0, 0.4)))
```

Predicted nodes and ranking:

```r
R> tm

<table>
<thead>
<tr>
<th>age</th>
<th>gender</th>
<th>q1</th>
<th>q2</th>
<th>q3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>male</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>female</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>female</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

R> predict(bt2, tm, type = "node")

```

```
1 2 3  
7 3 5 
```

```
R> predict(bt2, tm, type = function(object) t(rank(-worth(object))))

          Barbara  Anni  Hana Fiona  Mandy  Anja
1         1       6     5       4     3     2
2         2       3     1       4     5     6
3         3       4     1       2     6     5
```
Summary

- All new implementation of model-based recursive partitioning in *partykit*.
- Enables more efficient computations, rapid prototyping, flexible customization.
- Some new inference options.

