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

Achim Zeileis, Torsten Hothorn

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

- Motivation: Trees and leaves
- Model-based (MOB) recursive partitioning
  - Model estimation
  - Tests for parameter instability
  - Segmentation
  - Pruning
  - Local models
- Implementation in R
  - Building blocks: Parties, models, mobsters
  - Old implementation in party
  - All new implementation in partykit
- Application
  - Paired comparisons for Germany's Topmodel finalists
  - Bradley-Terry trees
  - Implementation from scratch
Motivation: Trees


- **Data models:** Stochastic models, typically parametric.
  → Classical strategy in statistics. Regression models are still the workhorse for many empirical analyses.

- **Algorithmic models:** Flexible models, data-generating process unknown.  → Still few applications in many fields, e.g., social sciences or economics.

**Classical example:** Trees, i.e., modeling of dependent variable $y$ by “learning” a recursive partition w.r.t explanatory variables $z_1, \ldots, z_l$. 
Motivation: Leaves

Key features:
1. Predictive power in nonlinear regression relationships.
2. Interpretability (enhanced by visualization), i.e., no “black box” methods.

Typically: Simple models for univariate $y$, e.g., mean.

Idea: More complex models for more complex $y$, e.g., regression models, multivariate normal model, item responses, etc.

Here: Synthesis of parametric data models and algorithmic tree models.

Goal: Fitting local models by partitioning of the sample space.
Recursive partitioning

Model-based (MOB) algorithm:

1. Fit the parametric model in the current subsample.
2. Assess the stability of the parameters across each partitioning variable $z_j$.
3. Split sample along the $z_j^*$ with strongest instability: Choose breakpoint with highest improvement of the model fit.
4. Repeat steps 1–3 recursively in the subsamples until some stopping criterion is met.
**Recursive partitioning**

**Example:** Logistic regression, assessing differences in the effect of “preferential treatment” (“women and children first”?) in the Titanic survival data.

**In R:** Generalized linear model tree with binomial family (and default logit link).

```r
R> mb <- glmtree(Survived ~ Treatment | Age + Gender + Class, + data = ttnc, family = binomial, alpha = 0.05, prune = "BIC")
R> plot(mb)
R> print(mb)
```

**Result:** Log-odds ratio of survival given treatment differs across classes (slope), as does the survival probability of male adults (intercept).
Recursive partitioning

Node 2 (n = 706)
Normal (Male&Adult)
Preferential (Female|Child)

Yes
No

Node 4 (n = 285)
Normal (Male&Adult)
Preferential (Female|Child)

Yes
No

Node 5 (n = 1210)
Normal (Male&Adult)
Preferential (Female|Child)
Recursive partitioning

Generalized linear model tree (family: binomial)

Model formula:
Survived ~ Treatment | Age + Gender + Class

Fitted party:
[1] root
  |   (Intercept) Treatment Preferential
  |   -1.641 1.327
  | [3] Class in 1st, 2nd, Crew
  |   |   (Intercept) Treatment Preferential
  |   |   -2.398 4.477
  |   | [5] Class in 1st, Crew: n = 1210
  |   |   (Intercept) Treatment Preferential
  |   |   -1.152 4.318

Number of inner nodes: 2
Number of terminal nodes: 3
Number of parameters per node: 2
Objective function (negative log-likelihood): 1061
1. Model estimation

**Models:** $\mathcal{M}(y, x, \theta)$ with (potentially multivariate) observations $y$, optionally regressors $x$, and $k$-dimensional parameter vector $\theta \in \Theta$.

**Parameter estimation:** $\hat{\theta}$ by optimization of additive objective function $\Psi(y, x, \theta)$ for $n$ observations $y_i (i = 1, \ldots, n)$:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} \Psi(y_i, x_i, \theta).$$

**Special cases:** Maximum likelihood (ML), weighted and ordinary least squares (OLS and WLS), quasi-ML, and other M-estimators.
1. Model estimation

**Estimating function:** \( \hat{\theta} \) can also be defined in terms of

\[
\sum_{i=1}^{n} \psi(y_i, x_i, \hat{\theta}) = 0,
\]

where \( \psi(y, x, \theta) = \frac{\partial \Psi(y, x, \theta)}{\partial \theta} \).

**Central limit theorem:** If there is a true parameter \( \theta_0 \) and given certain weak regularity conditions:

\[
\sqrt{n}(\hat{\theta} - \theta_0) \xrightarrow{d} \mathcal{N}(0, V(\theta_0)),
\]

where \( V(\theta_0) = \{A(\theta_0)\}^{-1} B(\theta_0) \{A(\theta_0)\}^{-1} \). A and B are the expectation of the derivative of \( \psi \) and the variance of \( \psi \), respectively.
1. Model estimation

Idea: In many situations, a single global model $\mathcal{M}(y, x, \theta)$ that fits all $n$ observations cannot be found. But it might be possible to find a partition w.r.t. the variables $z_1, \ldots, z_l$ so that a well-fitting model can be found locally in each cell of the partition.

Tools:

- Assess parameter instability w.r.t to partitioning variables $z_j \ (j = 1, \ldots, l)$.
- A general measure of deviation from the model is the estimating function $\psi(y, x, \theta)$. 
2. Tests for parameter instability

Generalized M-fluctuation tests capture instabilities in $\hat{\theta}$ for an ordering w.r.t $z_j$.

Basis: Empirical fluctuation process of cumulative deviations w.r.t. to an ordering $\sigma(z_{ij})$.

\[ W_j(t, \hat{\theta}) = \hat{B}^{-1/2} n^{-1/2} \sum_{i=1}^{\lfloor nt \rfloor} \psi(y_{\sigma(z_{ij})}, x_{\sigma(z_{ij})}, \hat{\theta}) \quad (0 \leq t \leq 1) \]

Functional central limit theorem: Under parameter stability $W_j(\cdot) \xrightarrow{d} W^0(\cdot)$, where $W^0$ is a $k$-dimensional Brownian bridge.
2. Tests for parameter instability

[Chart showing fluctuation process over time]
2. Tests for parameter instability

**Test statistics:** Scalar functional $\lambda(W_j)$ that captures deviations from zero.

**Null distribution:** Asymptotic distribution of $\lambda(W^0)$.

**Special cases:** Class of test encompasses many well-known tests for different classes of models. Certain functionals $\lambda$ are particularly intuitive for numeric and categorical $z_j$, respectively.

**Advantage:** Model $M(y, x, \hat{\theta})$ just has to be estimated once. Empirical estimating functions $\psi(y_i, x_i, \hat{\theta})$ just have to be re-ordered and aggregated for each $z_j$. 
2. Tests for parameter instability

Splitting numeric variables: Assess instability using supLM statistics.

\[
\lambda_{\text{supLM}}(W_j) = \max_{i=\tilde{i}, \ldots, \bar{i}} \left( \frac{i}{n} \cdot \frac{n - i}{n} \right)^{-1} \left\| W_j \left( \frac{i}{n} \right) \right\|_2^2.
\]

**Interpretation:** Maximization of single shift LM statistics for all conceivable breakpoints in \([\tilde{i}, \bar{i}]\).

**Limiting distribution:** Supremum of a squared, \(k\)-dimensional tied-down Bessel process.

**Potential alternatives:** Many other parameter instability tests from the same class of tests, e.g., a Cramér-von Mises test (or Nyblom-Hansen test), MOSUM tests, etc.
2. Tests for parameter instability

Splitting categorical variables: Assess instability using $\chi^2$ statistics.

$$\lambda_{\chi^2}(W_j) = \sum_{c=1}^{C} \frac{n}{|l_c|} \left\| \Delta_{l_c} W_j \left( \frac{i}{n} \right) \right\|_2^2.$$ 

**Feature:** Invariant for re-ordering of the $C$ categories and the observations within each category.

**Interpretation:** Capture instability for split-up into $C$ categories.

**Limiting distribution:** $\chi^2$ with $k \cdot (C - 1)$ degrees of freedom.
2. Tests for parameter instability

Splitting ordinal variables: Several strategies conceivable. Assess instability either as for categorical variables (if $C$ is low), or as for numeric variables (if $C$ is high), or via a specialized test.

$$
\lambda_{\text{maxLMo}}(W_j) = \max_{i \in \{i_1, \ldots, i_{C-1}\}} \left( \frac{i}{n} \cdot \frac{n-i}{n} \right)^{-1} \left\| W_j \left( \frac{i}{n} \right) \right\|_2^2,
$$

$$
\lambda_{\text{WDMo}}(W_j) = \max_{i \in \{i_1, \ldots, i_{C-1}\}} \left( \frac{i}{n} \cdot \frac{n-i}{n} \right)^{-1/2} \left\| W_j \left( \frac{i}{n} \right) \right\|_{\infty}.
$$

**Interpretation:** Assess only the possible splitpoints $i_1, \ldots, i_{C-1}$, based on $L_2$ or $L_\infty$ norm.

**Limiting distribution:** Maximum from selected points in a squared Bessel process or multivariate normal distribution, respectively.
3. Segmentation

**Goal:** Split model into \( b = 1, \ldots, B \) segments along the partitioning variable \( z_j \) associated with the highest parameter instability. Local optimization of

\[
\sum_{b} \sum_{i \in l_b} \psi(y_i, x_i, \theta_b).
\]

**Here:** Binary partitioning. Optionally, \( B = C \) can be chosen (without search) for categorical variables.
4. Pruning

**Goal:** Avoid overfitting.

**Pre-pruning:**
- Internal stopping criterium.
- Stop splitting when there is no significant parameter instability.
- Based on Bonferroni-corrected $p$ values of the fluctuation tests.

**Post-pruning:**
- Grow large tree (e.g., with high significance level).
- Prune splits that do not improve the model fit based on information criteria (e.g., AIC or BIC).

**Hyperparameters:** Significance level and information criterion penalty can be chosen manually (or possibly through cross-validation etc.).
Local models

Goals:
- Detection of interactions and nonlinearities in regressions.
- Add explanatory variables for models without regressors.
- Detect violations of parameter stability (measurement invariance) across several variables adaptively.

Mobsters:
- Linear and generalized linear model trees (Zeileis et al. 2008).
- Censored survival regression trees: parametric proportional hazard and accelerated failure time models (Zeileis et al. 2008).
- Beta regression trees (Grünn et al. 2012).
- Bradley-Terry trees for paired comparisons (Strobl et al. 2011).
- Item response theory (IRT) trees: Rasch, rating scale and partial credit model (Strobl et al. 2014, Abou El-Komboz et al. 2014).
Implementation: Building blocks

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

**Required functionality:**
- **Parties**: Class and methods for recursive partytions.
- **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 `mob()` in *party*

**Parties:** S4 class ‘BinaryTree’.
- Originally developed only for `ctree()` and somewhat “abused”.
- Rather rigid and hard to extend.

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

**Mobsters:**
- `mob()` already geared towards (generalized) linear models.
- Other interfaces in *psychotree* and *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 `mob()` in `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.

```
fit(y, x = NULL, start = NULL, weights = NULL, 
   offset = NULL, ...)
```

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

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

- `coef()`: Estimated parameters $\hat{\theta}$.
- `logLik()`: Maximized log-likelihood function $- \sum_i \Psi(y_i, x_i, \hat{\theta})$.
- `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 \( \sum_i \Psi(y_i, x_i, \hat{\theta}) \).
- **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.
Implementation: Parties

Class: ‘modelparty’ inheriting from ‘party’.

Main addition: Data handling for regressor and partitioning variables.
- The *Formula* package is used for two-part formulas, e.g.,
  \[ y \sim x_1 + x_2 \mid z_1 + z_2 + z_3. \]
- The corresponding terms are stored for the combined model and only for the partitioning variables.

Additional information: In info slots of ‘party’ and ‘partynode’.
- call, formula, Formula, terms (partitioning variables only), fit, control, dots, nreg.
- coefficients, objfun, object, nob, p.value, test.

Reusability: Could in principle be used for other model trees as well (inferred by other algorithms than MOB).
Bradley-Terry trees

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

Task: Preference scaling of attractiveness.

Data: Paired comparisons of attractiveness.

- Germany’s Next Topmodel 2007 finalists: Barbara, Anni, Hana, Fiona, Mandy, Anja.
- Survey with 192 respondents at Universität Tübingen.
- Available covariates: Gender, age, familiarity with the TV show.
- Familiarity assessed by yes/no questions: (1) Do you recognize the women?/Do you know the show? (2) Did you watch it regularly? (3) Did you watch the final show?/Do you know who won?
Bradley-Terry trees

**Model:** Bradley-Terry (or Bradley-Terry-Luce) model.

- Standard model for paired comparisons in social sciences.
- Parametrizes probability $\pi_{ij}$ for preferring object $i$ over $j$ in terms of corresponding “ability” or “worth” parameters $\theta_i$.

$$
\pi_{ij} = \frac{\theta_i}{\theta_i + \theta_j}
$$

$$
\logit(\pi_{ij}) = \log(\theta_i) - \log(\theta_j)
$$

- Maximum likelihood as a logistic or log-linear GLM.

**Mobster:** `bttree()` in `psychotree` (Strobl et al. 2011).

**Here:** Use `mob()` directly to build model from scratch using `btReg.fit()` from `psychotools`. 
Bradley-Terry trees

Node 1 (age)
- age ≤ 52
- age > 52

Node 2 (q2)
- q2 = 0.017
- yes
- no

Node 3 (n = 35)
- gender
- p = 0.007
- male
- female

Node 4
- gender
- p = 0.007
- male
- female

Node 5 (n = 71)

Node 6 (n = 56)

Node 7 (n = 30)
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
5.148 0.036  5.215
```
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
4.132   0.000   4.142
```
Bradley-Terry 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  0.08877  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
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)

Barbara | Anni  | Hana  | Fiona | Mandy
---------|-------|-------|-------|-------
3        | 1.3378| 1.23183| 2.0499| 0.8339| 0.6217
5        | 0.4387| 0.08877| 0.8463| 0.6942| -0.1000
6        | 0.9475| 0.72459| 0.4452| 0.6350| -0.4965
7        | 0.2178| -1.31663| -0.3059| -0.2591| -0.2357

Customization:

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

1. age
   p < 0.001
   ≤ 52
   > 52

2. q²
   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. male
   n = 71
   Estimated parameters:
   Barbara 0.43866
   Anni 0.08877
   Hana 0.84629
   Fiona 0.69424
   Mandy −0.10003

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

7. > 52
   n = 30
   Estimated parameters:
   Barbara 0.2178
   Anni −1.3166
   Hana −0.3059
   Fiona −0.2591
   Mandy −0.2357
Bradley-Terry trees

1. age
   - p < 0.001
     - ≤ 52
     - > 52

2. q2
   - p = 0.017
     - yes
       - Barbara: 0.19
         - Anni: 0.17
         - Hana: 0.39
         - Fiona: 0.11
         - Mandy: 0.09
         - Anja: 0.05
     - no

3. yes
   - Barbara: 0.19
   - Anni: 0.17
   - Hana: 0.39
   - Fiona: 0.11
   - Mandy: 0.09
   - Anja: 0.05

4. gender
   - p = 0.007
     - male
       - Barbara: 0.27
         - Anni: 0.21
         - Hana: 0.16
         - Fiona: 0.19
         - Mandy: 0.06
         - Anja: 0.10
     - female
       - Barbara: 0.26
         - Anni: 0.06
         - Hana: 0.15
         - Fiona: 0.16
         - Mandy: 0.16
         - Anja: 0.21
Bradley-Terry trees

3

5

6

7
Bradley-Terry trees

Apply plotting in all terminal nodes:

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> tm

<table>
<thead>
<tr>
<th></th>
<th>age</th>
<th>gender</th>
<th>q1</th>
<th>q2</th>
<th>q3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>60</td>
<td>male</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>female</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>35</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))))

<table>
<thead>
<tr>
<th>Barbara</th>
<th>Anni</th>
<th>Hana</th>
<th>Fiona</th>
<th>Mandy</th>
<th>Anja</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>
Summary

- Synthesis of parametric data models and algorithmic tree models.
- Based on modern class of parameter instability tests.
- Aims to minimize clearly defined objective function by greedy forward search.
- Can be applied general class of parametric models.
- Alternative to traditional means of model specification, especially for variables with unknown association.
- All new implementation in partykit.
- Enables more efficient computations, rapid prototyping, flexible customization.
References

Software:


Inference:


Trees:


