Model-Based Recursive Partitioning

Achim Zeileis

http://statmath.wu-wien.ac.at/~zeileis/

Overview

- Motivation: Trees and leaves
- Methodology
  - Model estimation
  - Tests for parameter instability
  - Segmentation
  - Pruning
- Applications
  - Costly journals
  - Beautiful professors
  - Choosey students
- Software

Motivation: Trees

Breiman (2001, *Statistical Science*) distinguishes two cultures of statistical modeling.

- **Data models:** Stochastic models, typically parametric.
- **Algorithmic models:** Flexible models, data-generating process unknown.

Example: Recursive partitioning models dependent variable Y by "learning" a partition w.r.t explanatory variables $Z_1, \dots, Z_l$.

Key features:

- Predictive power in nonlinear regression relationships.
- Interpretability (enhanced by visualization), i.e., no "black box" methods.

Motivation: Leaves

Typically: Simple models for univariate Y, e.g., mean or proportion.

Examples: CART and C4.5 in statistical and machine learning, respectively.

Idea: More complex models for multivariate Y, e.g., multivariate normal model, regression models, etc.

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

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

Recursive partitioning

Base algorithm:

1. Fit model for Y.
2. Assess association of Y and each 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 sub-samples until some stopping criterion is met.

Here: Segmentation (3) of parametric models (1) with additive objective function using parameter instability tests (2) and associated statistical significance (4).

1. Model estimation

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

Parameter estimation: $\widehat{\theta}$ by optimization of objective function $\Psi(Y, \theta)$ for n observations Y_i ($i = 1, \dots, n$):

$$\widehat{\theta} \quad = \quad \underset{\theta \in \Theta}{\text{argmin}} \sum_{i=1}^{n} \Psi(Y_i, \theta).$$

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

Central limit theorem: If there is a true parameter θ_0 and given certain weak regularity conditions, $\hat{\theta}$ is asymptotically normal with mean θ_0 and sandwich-type covariance.

1. Model estimation

Estimating function: $\widehat{\theta}$ can also be defined in terms of

$$\sum_{i=1}^{n} \psi(Y_i, \widehat{\theta}) = 0,$$

where $\psi(Y, \theta) = \partial\Psi(Y, \theta)/\partial\theta$.

Idea: In many situations, a single global model $\mathcal{M}(Y, \theta)$ that fits **all** n observations cannot be found. But it might be possible to find a partition w.r.t. the variables $Z = (Z_1, \dots, Z_l)$ so that a well-fitting model can be found locally in each cell of the partition.

Tool: Assess parameter instability w.r.t to partitioning variables $Z_j \in \mathcal{Z}_j$ ($j = 1, \dots, l$).

2. Tests for parameter instability

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

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

$$W_j(t, \widehat{\theta}) \quad = \quad \widehat{B}^{-1/2} n^{-1/2} \sum_{i=1}^{\lfloor nt \rfloor} \psi(Y_{\sigma(Z_{ij})}, \widehat{\theta}) \qquad (0 \le t \le 1)$$

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

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 λ are particularly intuitive for numeric and categorical Z_j, respectively.

Advantage: Model $\mathcal{M}(Y, \widehat{\theta})$ just has to be estimated once. Empirical estimating functions $\psi(Y_i, \widehat{\theta})$ 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.

$$\lambda_{\sup LM}(W_j) \quad = \quad \max_{i=\underline{\imath},\dots,\overline{\imath}} \left(\frac{i}{n} \cdot \frac{n-i}{n}\right)^{-1} \left|\left| W_j\left(\frac{i}{n}\right)\right|\right|_2^2 .$$

Interpretation: Maximization of single shift LM statistics for all conceivable breakpoints in $[\underline{\imath}, \overline{\imath}]$.

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

2. Tests for parameter instability

Splitting categorical variables: Assess instability using χ^2 statistics.

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

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

Interpretation: Captures instability for split-up into C categories.

Limiting distribution: χ^2 with $k \cdot (C-1)$ degrees of freedom.

3. Segmentation

Goal: Split model into $b = 1, \dots, B$ segments along the partitioning variable Z_j associated with the highest parameter instability. Local optimization of

$$\sum_b \sum_{i \in I_b} \Psi(Y_i, \theta_b).$$

$B = 2$: Exhaustive search of order $O(n)$.

$B > 2$: Exhaustive search is of order $O(n^{B-1})$, but can be replaced by dynamic programming of order $O(n^2)$. Different methods (e.g., information criteria) can choose B adaptively.

Here: Binary partitioning.

4. Pruning

Pruning: Avoid overfitting.

Pre-pruning: Internal stopping criterion. Stop splitting when there is no significant parameter instability.

Post-pruning: Grow large tree and prune splits that do not improve the model fit (e.g., via cross-validation or information criteria).

Here: Pre-pruning based on Bonferroni-corrected *p* values of the fluctuation tests.

Costly journals

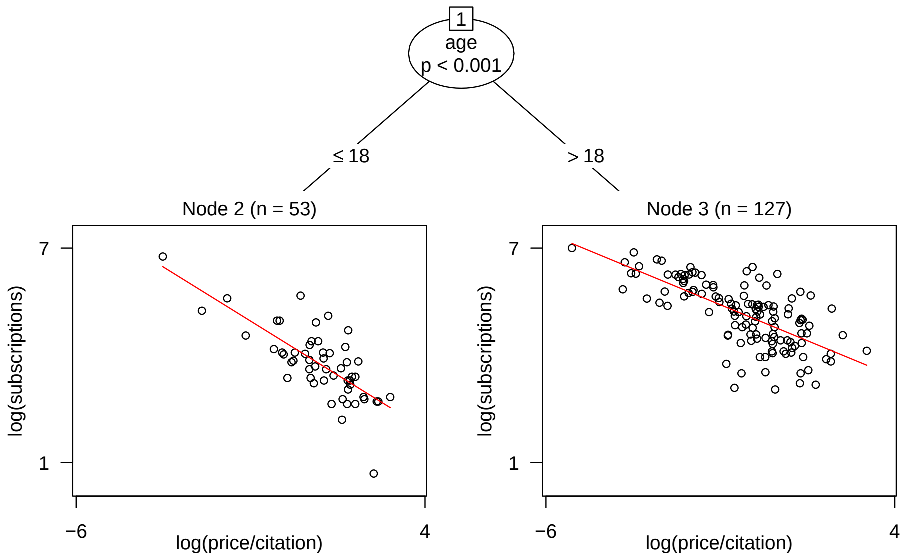
Task: Price elasticity of demand for economics journals.

Source: Bergstrom (2001, *Journal of Economic Perspectives*) “Free Labor for Costly Journals?”, used in Stock & Watson (2007), *Introduction to Econometrics*.

Model: Linear regression via OLS.

- Demand: Number of US library subscriptions.
- Price: Average price per citation.
- Log-log-specification: Demand explained by price.
- Further variables without obvious relationship: Age (in years), number of characters per page, society (factor).

Costly journals



Costly journals

Recursive partitioning:

	Regressors		Partitioning variables				
	(Const.)	log(Pr./Cit.)	Price	Cit.	Age	Chars	Society
1	4.766	−0.533	3.280	5.261	42.198	7.436	6.562
	< 0.001	< 0.001	0.660	0.988	< 0.001	0.830	0.922
2	4.353	−0.605	0.650	3.726	5.613	1.751	3.342
	< 0.001	< 0.001	0.998	0.998	0.935	1.000	1.000
3	5.011	−0.403	0.608	6.839	5.987	2.782	3.370
	< 0.001	< 0.001	0.999	0.894	0.960	1.000	1.000

(Wald tests for regressors, parameter instability tests for partitioning variables.)

Beautiful professors

Task: Correlation of beauty and teaching evaluations for professors.

Source: Hamermesh & Parker (2005, *Economics of Education Review*). “Beauty in the Classroom: Instructors’ Pulchritude and Putative Pedagogical Productivity.”

Model: Linear regression via WLS.

- Response: Average teaching evaluation per course (on scale 1–5).
- Explanatory variables: Standardized measure of beauty and factors gender, minority, tenure, etc.
- Weights: Number of students per course.

Beautiful professors

	All	Men	Women
(Constant)	4.216	4.101	4.027
Beauty	0.283	0.383	0.133
Gender (= w)	−0.213		
Minority	−0.327	−0.014	−0.279
Native speaker	−0.217	−0.388	−0.288
Tenure track	−0.132	−0.053	−0.064
Lower division	−0.050	0.004	−0.244
R^2	0.271	0.316	

(Remark: Only courses with more than a single credit point.)

Beautiful professors

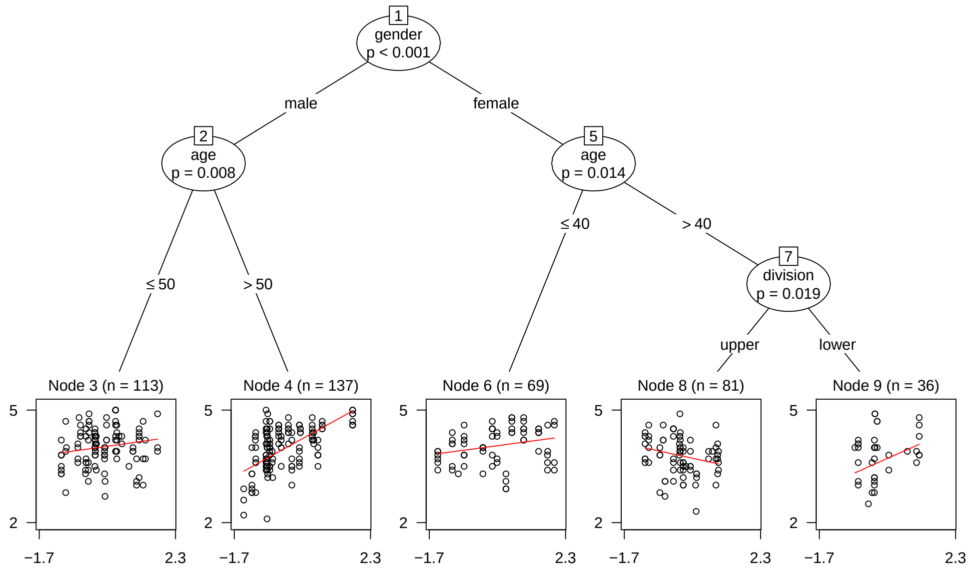
Hamermesh & Parker:

- Model with all factors (main effects).
- Improvement for separate models by gender.
- No association with age (linear or quadratic).

Here:

- Model for evaluation explained by beauty.
- Other variables as partitioning variables.
- Adaptive incorporation of correlations and interactions.

Beautiful professors



Beautiful professors

Recursive partitioning:

	(Const.)	Beauty
3	3.997	0.129
4	4.086	0.503
6	4.014	0.122
8	3.775	−0.198
9	3.590	0.403

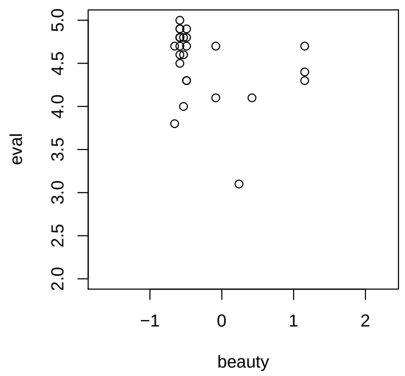
Model comparison:

Model	R^2	Parameters
full sample	0.271	7
nested by gender	0.316	12
recursively partitioned	0.382	10 + 4

Beautiful professors

Single credit courses:

- Different type of courses: Yoga, aerobic, etc.
- Associated with second strongest instability (after gender).
- Sub-samples too small for separated models: 18 (m), 9 (f).



Choosy students

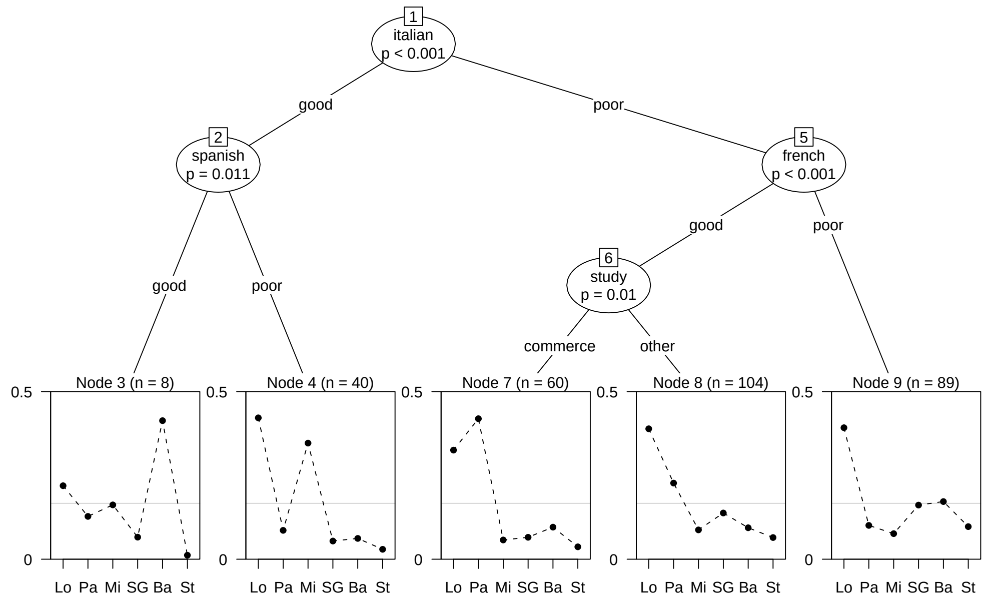
Task: Choice of university in student exchange programmes.

Source: Dittrich, Hatzinger, Katzenbeisser (1998, *Journal of the Royal Statistical Society C*). “Modelling the Effect of Subject-Specific Covariates in Paired Comparison Studies with an Application to University Rankings.”

Model: Paired comparison via Bradley-Terry(-Luce).

- Ranking of six european management schools: London (LSE), Paris (HEC), Milano (Luigi Bocconi), St. Gallen (HSG), Barcelona (ESADE), Stockholm (HHS).
- Interviews with about 300 students from WU Wien.
- Additional information: Gender, studies, foreign language skills.

Choosy students



Choosy students

Recursive partitioning:

	London	Paris	Milano	St. Gallen	Barcelona	Stockholm
3	0.22	0.13	0.16	0.07	0.41	0.01
4	0.42	0.09	0.35	0.05	0.06	0.03
7	0.33	0.42	0.06	0.07	0.10	0.04
8	0.39	0.23	0.09	0.14	0.09	0.06
9	0.39	0.10	0.08	0.16	0.17	0.10

(Standardized ranking from Bradley-Terry model.)

Software

Implementation: In R system for statistical computing.

- Object-oriented implementation of model-based recursive partitioning in function `mob()` from package **party**.
- Underlying inference methods in package **strucchange**.
- Convenient interfaces for linear regression (`lm.fit()`), generalized linear models (`glm.fit()`), and survival regression (`survreg()`) are readily available.
- Currently: Hand-crafted code for Bradley-Terry model (interfacing `glm.fit()`), not in package.

Software

Extension requirements:

- S4 “`StatModel`” objects (**modeltools** package): Separate data handling (in particular, formula processing) from model fitting.
- Fitted models must provide methods: `estfun()`, `weights()`, `reweight()` (at least for 0/1 weights), and extractor for objective function (default: `deviance()`).
- Further methods are re-used (if available): `print()`, `predict()`, `coef()`, `summary()`, `residuals()`, `logLik()`.

Easy if: Model already available in R with

- Fitted model class with all the usual extractor functions.
- Access to empirical estimating functions (`estfun()` method).
- In addition to formula interface (à la `lm()`): Fitting function (à la `lm.fit()`) that returns sufficiently post-processed output.

Software

Caveats:

- For visualization: Panel-generating function for **grid** graphics.
- `mob()` interprets `weights` as case weights (and expects the “`StatModel`” objects to do the same).
- Non-standard formula processing for multivariate responses.
- Hopefully: New model/formula interface soon on R-Forge.

Example: Simple implementation of basic Bradley-Terry model.

- Interfaces: `btl()` and `btl.fit()` plus methods.
- Workhorse: Set up design matrix, call `glm.fit()` with `family = binomial()`, suitably aggregate results.
- Glue code: S4 “`BTL`” object with few additional methods.

Implementation of simple Bradley-Terry models

Artificial data:

```
R> pc <- rbind(
+    c(1, 1, 1), # a > b > c
+    c(1, 1, 0), # a > c > b
+    c(1, 0, 0), # c > a > b
+    c(1, 1, 1)  # a > b > c
+  )
R> colnames(pc) <- c("ab", "ac", "bc")
```

Question: Proper data structures for paired comparison data?

Ideally: `pc` should be treated like a *vector* of length 4 (# subjects) with suitable meta-data that reflects # objects, labels, printing, etc.

Implementation of simple Bradley-Terry models

Interfaces: Formula interface and workhorse fitting function.

```
R> btl(pc ~ 1)

Bradley-Terry-Luce model

Coefficients:
      a       b
 1.7542 -0.4158

Standardized latent ranking:
     a      b      c
0.7769 0.0887 0.1344

R> pc_btl <- btl.fit(pc)
R> class(pc_btl)

[1] "btl"
```

Implementation of simple Bradley-Terry models

```
R> coef(pc_btl)

         a          b
 1.7541765 -0.4158042

R> coef(pc_btl, log = FALSE)

         a          b          c
0.77686224 0.08870199 0.13443577

R> estfun(pc_btl)

              a          b
[1,]  0.2500030  0.4999987
[2,]  0.2500030 -0.5000014
[3,] -0.7500083 -0.5000014
[4,]  0.2500030  0.4999987

R> deviance(pc_btl)

[1] 11.36700

R> logLik(pc_btl)

'log Lik.' -5.683498 (df=2)
```

Implementation of simple Bradley-Terry models

```
R> btl.fit(y = pc, weights = c(1, 1, 1, 0))

Bradley-Terry-Luce model

Coefficients:
     a      b
 1.145 -1.145

Standardized latent ranking:
      a       b       c
0.70450 0.07133 0.22417
```

Implementation of simple Bradley-Terry models

Interface: "`StatModel`" glue code.

```
R> class(BTL)

[1] "StatModel"
attr(,"package")
[1] "modeltools"

R> mf <- ModelEnvFormula(ab + ac + bc ~ 1,
+    data = as.data.frame(pc))
R> pc_BTL <- fit(BTL, mf)
R> class(pc_BTL)

[1] "BTL" "btl"

R> pc_BTL

BTL coefficients:
      a       b
 1.7542 -0.4158
```

Implementation of simple Bradley-Terry models

```
R> pc_BTL2 <- reweight(pc_BTL, weights = c(1, 1, 1, 0))
R> weights(pc_BTL2)

[1] 1 1 1 0

R> coef(pc_BTL2)

        a         b
 1.145071 -1.145071

R> estfun(pc_BTL2)

              a          b
[1,]  0.3333340  0.6666672
[2,]  0.3333340 -0.3333340
[3,] -0.6666672 -0.3333340
[4,]  0.0000000  0.0000000
```

Implementation of simple Bradley-Terry models

```
R> load("cems.rda")
R> cems <- cems[!apply(sapply(cems[,1:15], is.na), 1, all),]

R> cems_mob <- mob(ab + ac + ad + ae + af + bc + bd + be +
+    bf + cd + ce + cf + de + df + ef ~ 1 | study + english +
+    french + spanish + italian + work + gender + intdegree,
+    data = cems, model = BTL, na.action = na.pass,
+    control = mob_control(minsplit = 5))

R> plot(cems_mob, terminal_panel = node_btlplot,
+    tnex = 2, tp_args = list(yscale = c(0, 0.5),
+    names = c("Lo", "Pa", "Mi", "SG", "Ba", "St")))

R> coef(cems_mob)

         a          b          c         d         e
3 2.915557 2.37530103  2.6132469 1.7116796 3.5496433
4 2.657933 1.06913836  2.4611060 0.6068164 0.7413380
7 2.174962 2.42810393  0.4400174 0.5704167 0.9507593
8 1.794987 1.25646206  0.3024828 0.7576933 0.3729114
9 1.394938 0.03678015 -0.2427147 0.5071161 0.5702408
```

Summary

Model-based recursive partitioning:

- Synthesis of classical 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.
- Object-oriented implementation freely available: Extension for new models requires some coding but is limited if interfaced model is well designed.