Journal of Machine Learning Research 16 (2015) 3905-3909 Submitted 3/14; Revised 1/15; Published 12/15

# *partykit*: A Modular Toolkit for Recursive Partytioning in R

**Torsten Hothorn** Torsten.Hothorn@R-project.org
*Institut für Epidemiologie, Biostatistik und Prävention, Universität Zürich*

**Achim Zeileis** Achim.Zeileis@R-project.org
*Institut für Statistik, Universität Innsbruck*

**Editor:** Cheng Soon Ong

## Abstract

The R package *partykit* provides a flexible toolkit for learning, representing, summarizing, and visualizing a wide range of tree-structured regression and classification models. The functionality encompasses: (a) basic infrastructure for *representing* trees (inferred by any algorithm) so that unified `print`/`plot`/`predict` methods are available; (b) dedicated methods for trees with *constant fits* in the leaves (or terminal nodes) along with suitable coercion functions to create such trees (e.g., by *rpart*, *RWeka*, PMML); (c) a reimplementation of *conditional inference trees* (`ctree`, originally provided in the *party* package); (d) an extended reimplementation of *model-based recursive partitioning* (`mob`, also originally in *party*) along with dedicated methods for trees with parametric models in the leaves. Here, a brief overview of the package and its design is given while more detailed discussions of items (a)–(d) are available in vignettes accompanying the package.

**Keywords:** recursive partitioning, regression trees, classification trees, statistical learning, R

## 1. Overview

In the more than fifty years since Morgan and Sonquist (1963) published their seminal paper on "automatic interaction detection", a wide range of methods has been suggested that is usually termed "recursive partitioning" or "decision trees" or "tree(-structured) models" etc. The particularly influential algorithms include CART (classification and regression trees, Breiman et al., 1984), C4.5 (Quinlan, 1993), QUEST/GUIDE (Loh and Shih, 1997; Loh, 2002), and CTree (Hothorn et al., 2006) among many others (see Loh, 2014, for a recent overview). Reflecting the heterogeneity of conceptual algorithms, a wide range of computational implementations in various software systems emerged: Typically the original authors of an algorithm also provide accompanying software but many software systems, including *Weka* (Witten and Frank, 2005) or R (R Core Team, 2014), also provide collections of various types of trees. Within R the list of prominent packages includes *rpart* (Therneau and Atkinson, 1997, implementing CART), *RWeka* (Hornik et al., 2009, with interfaces to J4.8, M5', LMT from *Weka*), and *party* (Hothorn et al., 2015, implementing CTree and MOB) among many others. See the CRAN task view "Machine Learning" (Hothorn, 2014) for an overview.

All of these algorithms and software implementations have to deal with similar challenges. However, due to the fragmentation of the communities in which they are published – ranging from statistics over machine learning to various applied fields – many discussions of the algorithms do not reuse established theoretical results and terminology. Similarly, there is no common "language" for the software implementations and different solutions are provided by different packages (even within R) with relatively little reuse of code. The *partykit* aims at mitigating the latter issue by providing a common unified infrastructure for recursive partytioning in the R system for statistical computing. In particular, *partykit* provides tools for representing, printing, plotting trees and computing predictions. The design principles are:

- One 'agnostic' base class ('`party`') encompassing a very wide range of different tree types.
- Subclasses for important types of trees, e.g., trees with constant fits ('`constparty`') or with parametric models ('`modelparty`') in each terminal node (or leaf).
- Nodes are recursive objects, i.e., a node can contain child nodes.
- Keep the (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 object classes in R (`rpart`, `J48`, etc.) to the new class.
- Usage of simple/fast S3 classes and methods.

In addition to all of this generic infrastructure, two specific tree algorithms are implemented in *partykit* as well: `ctree` for conditional inference trees (Hothorn et al., 2006) and `mob` for model-based recursive partitioning (Zeileis et al., 2008).

## 2. Installation and Documentation

The *partykit* package is an add-on package for the R system for statistical computing. It is available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=partykit` and can be installed from within R, e.g., using `install.packages`. It depends on R (at least 2.15.0) as well as the base packages *graphics*, *grid*, *stats*, and the recommended *survival*. Furthermore, various suggested packages are needed for certain special functionalities in the package. To install all of these required and suggested packages in one go, the command `install.packages("partykit", dependencies = TRUE)` can be used.

In addition to the stable release version on CRAN, the current development release can be installed from R-Forge (Theußl and Zeileis, 2009). In addition to source and binary packages the entire version history is available through R-Forge's *Subversion* source code management system.

Along with the package extensive documentation with examples is shipped. The manual pages provide basic technical information on all functions while much more detailed descriptions along with hands-on examples are provided in the four package vignettes. First, the vignette `"partykit"` introduces the basic '`party`' class and associated infrastructure while three further vignettes discuss the tools built on top of it: `"constparty"` covers the eponymous class (as well as the simplified '`simpleparty`' class) for constant-fit trees along with suitable coercion functions, and `"ctree"` and `"mob"` discuss the new `ctree` and `mob` implementations, respectively. Each of the vignettes can be viewed within R via `vignette(`*"name"*`, package = "partykit")` and the underlying source code (in R with LaTeX text) is also available in the source package.

## 3. User Interface

The *partykit* package provides functionality at different levels. First, there is basic infrastructure for representing, modifying, and displaying trees and recursive partitions – these tools are mostly intended for developers and described in the next section. Second, there are tools for inferring trees from data or for importing trees inferred by other software into *partykit*.

While originally an important goal for the development of *partykit* was to provide infrastructure for the authors' own tree induction algorithms CTree and MOB, the design was very careful to separate as much functionality as possible into more general classes that are useful for a far broader class of trees. In particular, to be able to print/plot/predict different trees in a unified way, there

| Algorithm | Software implementation | Object class | Original reference |
|---|---|---|---|
| CART/RPart | `rpart::rpart` + `as.party` | `constparty` | Breiman et al. (1984) |
| C4.5/J4.8 | *Weka*/`RWeka::J48` + `as.party` | `constparty` | Quinlan (1993) |
| QUEST | SPSS/*AnswerTree* + `pmmlTreeModel` | `simpleparty` | Loh and Shih (1997) |
| CTree | `ctree` | `constparty` | Hothorn et al. (2006) |
| MOB | `mob`, `lmtree`, `glmtree`, ... | `modelparty` | Zeileis et al. (2008) |
| EvTree | `evtree::evtree` | `constparty` | Grubinger et al. (2014) |

Table 1: Selected implementations of tree algorithms that can be interfaced through *partykit*. The second column lists external software, R functions from other packages (with `::` syntax) and from *partykit*.

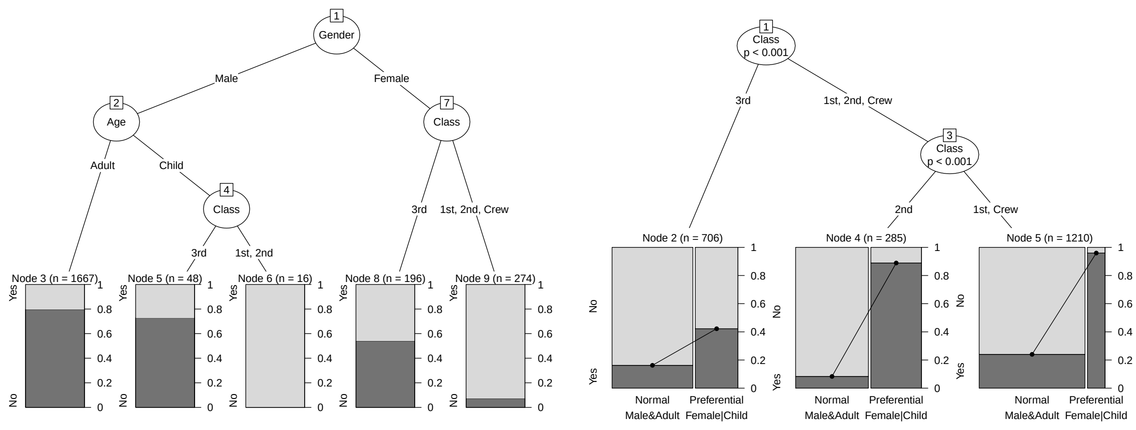

Figure 1: Tree visualizations of survival on Titanic: '`rpart`' tree converted with `as.party` and visualized by *partykit* (left); and logistic-regression-based tree fitted by `glmtree` (right).

are so-called coercion functions for transforming trees learned in other software packages (inside and outside of R) to the classes provided by *partykit*. Specifically, tree objects learned by `rpart` (Therneau and Atkinson, 1997, implementing CART, Breiman et al., 1984) and by `J48` from *RWeka* (Hornik et al., 2009, interfacing *Weka*'s J4.8 algorithm for C4.5, Quinlan, 1993) can be coerced by `as.party` to the same object class '`constparty`'. This is a general class that can in principle represent all the major classical tree types with constant fits in the terminal nodes. Also, the same class is employed for conditional inference trees (CTree) that can be learned with the `ctree` function directly within *partykit* or evolutionary trees from package *evtree* (Grubinger et al., 2014).

Not only trees learned within R can be transformed to the proposed infrastructure but also trees from other software packages. Either a dedicated interface has to be created using the building blocks described in the next section (e.g., as done for the J4.8 tree in *RWeka*) or PMML (Predictive Model Markup Language) can be used as an intermediate exchange format. This is an XML standard created by an international consortium (Data Mining Group, 2014) that includes a `<TreeModel>` tag with support for constant-fit classification and regression trees. The function `pmmlTreeModel` allows to read these files and represents them as '`simpleparty`' objects in *partykit*. The reason for not using the '`constparty`' class as above is that the PMML format only stores point predictions (e.g., a mean or proportion) rather than all observations from the learning sample. So far, the PMML interface has been tested with output from the R package *pmml* and SPSS's *AnswerTree* model. The latter includes an implementation of the QUEST algorithm (Loh and Shih, 1997).

Finally, the *partykit* function `mob` implements model-based recursive partitioning (MOB) along with "mobster" interfaces for certain models (e.g., `lmtree`, `glmtree`). These return objects of class '`modelparty`' where nodes are associated with statistical models (as opposed to simple constant fits). In principle, this may also be adapted to other model trees (such as GUIDE, LMT, or M5') but no such interface is currently available.

All of these different trees (see Table 1 for an overview) use the same infrastructure at the core but possibly with different options enabled. In all cases, the functions `print`, `plot`, and `predict` can be used to create textual and graphical displays of the tree and for computing predictions on new data, respectively. As an example for the visualizations, Figure 1 shows two different trees fitted to the well-known data on survival of passengers on the ill-fated maiden voyage of the RMS Titanic: The left panel shows a CART tree with constant fits learned by *rpart* and converted to *partykit*. The right panel shows a MOB tree learned with *partykit* with a logistic regression for treatment effects in the terminal nodes. Additionally, the are further utility functions, e.g., `nodeapply` can be employed to access further information stored in the nodes of a tree and `nodeprune` can prune selected nodes.

## 4. Developer Infrastructure

The unified infrastructure at the core of *partykit* is especially appealing for developers who either want to implement new tree algorithms or represent trees learned in other systems.

Here, we briefly outline the most important classes and refer to the vignettes for more details:

**'`partysplit`':** Split with integer ID for the splitting variable, breakpoint(s), indexes for the kids.

**'`partynode`':** Node specification with integer ID, a '`partysplit`', and a list of kids (if any) that are '`partynode`' objects again.

**'`party`':** Tree with a recursive '`partynode`' and a '`data.frame`' (optionally empty), potentially plus information about fitted values and '`terms`' allowing to preprocess new data for predictions.

All classes have an additional slot for storing arbitrary information at any level of the tree. This is exploited by '`constparty`', '`simpleparty`', and '`modelparty`' which store the observed response, point predictions, and fitted parametrics models, respectively.

## 5. Discussion and Outlook

Package *partykit* provides a toolkit for trees in R that gives emphasis to flexibility and extensibility. The infrastructure is easily accessible and accompanied by detailed manual pages and package vignettes. The package facilitates the implementation of new algorithms or interfacing other software by providing common building blocks for computing on trees (representation, printing, plotting, predictions, etc.). Using these building blocks developers of tree software can focus on implementing the learning algorithm (selection of variables and split points, stopping criteria, pruning, etc.). The package also provides functions for inferring trees where the computationally intensive parts are either in C (`ctree`) or employ R's fitting functions (`mob`). The simple and lean base classes that separate data and tree structure are also appealing for storing forests – a first proof-of-concept reimplementation of `cforest` is in the package with further extension planned. Users and developers that have questions or comments about the package can either contact the maintainers or use the forum on R-Forge at `https://R-Forge.R-project.org/forum/forum.php?forum_id=852`.

## Acknowledgments

We are thankful to the organizers and participants of the "Workshop on Classification and Regression Trees" (March 2014), sponsored by the Institute for Mathematical Sciences of the National University of Singapore, for helpful feedback and stimulating discussions.

## References

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, California, 1984.

Data Mining Group. Predictive model markup language, 2014. URL `http://www.dmg.org/`. Version 4.2.

Thomas Grubinger, Achim Zeileis, and Karl-Peter Pfeiffer. *evtree*: Evolutionary learning of globally optimal classification and regression trees in R. *Journal of Statistical Software*, 61(1), 1–29 2014.

Kurt Hornik, Christian Buchta, and Achim Zeileis. Open-source machine learning: R meets *Weka*. *Computational Statistics*, 24(2):225–232, 2009.

Torsten Hothorn. CRAN task view: Machine learning & statistical learning, 2014. URL `http://CRAN.R-project.org/view=MachineLearning`. Version 2014-12-18.

Torsten Hothorn, Kurt Hornik, and Achim Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.

Torsten Hothorn, Kurt Hornik, Carolin Strobl, and Achim Zeileis. *party: A Laboratory for Recursive Partytioning*, 2015. URL `http://CRAN.R-project.org/package=party`. R package version 1.0-20.

Wei-Yin Loh. Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica*, 12(2):361–386, 2002.

Wei-Yin Loh. Fifty years of classification and regression trees. *International Statistical Review*, 82 (3):329–348, 2014.

Wei-Yin Loh and Yu-Shan Shih. Split selection methods for classification trees. *Statistica Sinica*, 7 (4):815–840, 1997.

James N. Morgan and John A. Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, 58(302):415–434, 1963.

John R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, 1993.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014. URL `http://www.R-project.org/`.

Terry M. Therneau and Elizabeth J. Atkinson. An introduction to recursive partitioning using the *rpart* routine. Technical Report 61, Section of Biostatistics, Mayo Clinic, Rochester, 1997. URL `http://www.mayo.edu/hsr/techrpt/61.pdf`.

Stefan Theußl and Achim Zeileis. Collaborative software development using R-Forge. *The R Journal*, 1(1):9–14, May 2009.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

Achim Zeileis, Torsten Hothorn, and Kurt Hornik. Model-based recursive partitioning. *Journal of Computational and Graphical Statistics*, 17(2):492–514, 2008.