Review of: Let's Have a party ! An Open-Source Toolbox for Recursive Partytioning
Torsten Hothorn and Achim Zeileis and Kurt Hornik

I think this a well-conceived piece of software. However, the paper leaves me puzzled about some crucial aspects of what it offers, and of its adaptability. The paper requires rewriting to place more emphasis on its usefulness as a toolbox. Obviously, there is a severe limit to what can be provided in four pages, and it is broadbrush information that is required.

Goal: The aim of the package is (I think) to provide an infrastructure for tree-based learners, functioning in particular as a framework for implementing specific algorithms that have been developed by these authors. Different documents do however give accounts that are not totally straightforward to reconcile; see below.

Evaluation: As far as I can judge, the authors have done a good job in providing a basic framework, with the R help files and pdf documents that are supplied with the package elaborating on some of the important detail. Implementation in an R package structure is a good starting point for improving and adapting the software. Also, all of R's extensive data manipulation and graphics abilities are immediately available to package users, and facilitates comparison with other tree-based learners that are available as R packages. These are large advantages over free-standing computer software.

Insightfulness of Description: The description in the paper left a number of questions unanswered. A better use could, I believe, be made of the four pages. Given that this is a toolbox, surely the emphasis should be on how the toolbox is used, or can be used, to implement specific methods. The examples of the specific implementations are good, but given the limitations of space, surely better insight on the framework is the higher priority. How, for example, are the items in the toolboox put together to create **ctree**? How might they be put together to create a direct analogue of **rpart**? Surely it is possible to give broadbrush information. There should be direct reference to web-based materials that give more detailed information and examples.

Accompanying the package is a vignette (pdf file that has the title: "party: a Laboratory for Recursive Part(y)itioning". This gave some insight. I could not find anything that gave much guidance to the underlying C code; such documents as I could find in the inst/documentation subdirectory was not very enlightening, to me at least.

A Unified Framework: The vignette just noted states that the package presents "a unified framework embedding recursive binary partitioning into the well-defined theory of permutation tests developed by Strasser and Weber (1999)". There is no mention of this "unified framework" for permutation tests in the present document, where the emphasis is that "party provides basic classes and methods for recursive partitioning". Certainly there is a mention of "permutation tests for split selection", but this is with respect to the specific **ctree** method, not the framework. How deeply embedded is the use of a permutation test methodology?

There is the further point that local optimality in some hypothesis testing sense is not guaranteed (in general will not) lead to global optimality. What is relevant here is sequential testing, not multiple testing per se. Do permutation methods replace the use of the complexity parameter and cross-validation as in **rpart**, or the bootstrap OOB accuracy measure as in **randomForest**?

The specific tree-based methods that have been implemented to date are the authors' own, i.e. **ctree** (how close is this to e.g. **rpart** in the **rpart** package?), and **cforest**. How easy would it be to implement a close analogue of **rpart**, or of r**andomForest**? An outline of the steps needed, and the main code components, would be helpful.

Comparison to Previous Implementations: There is no discussion of run-time or memory considerations. In my own quick comparison between the **randomForest** package and **cforest** in the present package, **cforest** was slower by a factor of about 10. As these authors are providing something quite different to randomForest and other such software, i.e., a basic toolkit that is under continuing development, I do not regard this as a serious criticism. The code will no doubt be worked on to become, in time, more efficient. It may be a criticism of these authors' specific use of permutation methods, if the use of permutation methods does indeed work to slow the calculations substantially.

In any case, if my experience of the relative timings is at all typical, potential users should be warned that calculations may be substantially slower than in other implementations of tree-based methods.

Future Development: What plans are there for further development of the package? Improving documentation for its use as a toolbox would seem a high priority.

Detailed Comments

One, or two exceptions?: In the final two sentences of the first paragraph, are the authors really saying that there are two notable exceptions? Or what is the intent of these sentences? R is surely far and away the more comprehensive. Additionally it should be noted that Weka has been interfaced to R.

Unbiased variable importance measures?: I take issue with the claim (page 1, line -4) that **cforest** "can provide unbiased variable importance measures". There are indeed multiple measures. Whichever of these measures is taken, the estimate is unbiased only with respect to a very specific sampling model.

Automatic Testing of R Packages: The automatic testing at the end of Section 3 is mainly a check that the code and associated documentation meets certain programming and consistency standards. It does not guarantee that results will be correct, and is in that sense a very limited form of testing.