Journal of Statistical Software

May 2005, Volume 14, Issue 1. *http://www.jstatsoft.org/*

pinktoe: Semi-automatic Traversal of Trees

Guy Nason
University of Bristol

Abstract

Tree based methods in S or R are extremely useful and popular. For simple trees and memorable variables it is easy to predict the outcome for a new case using only a standard decision tree diagram. However, for large trees or trees where the variable description is complex the decision tree diagram is often not enough. This article describes **pinktoe**: an R package containing two tools to assist with the semi-automatic traversal of trees. The `PT` tool creates a widget for each node to be visited in the tree that is needed to make a decision and permits the user to make decisions using radiobuttons. The `pinktoe` function generates a suite of HTML and Perl files that permit a CGI-enabled website to issue step-by-step questions to a user wishing to make a prediction using a tree.

Keywords: CART, tree traversal, web trees, `"rpart"` objects.

1. Introduction

pinktoe is an R package that simplifies the decision making process when using tree objects: especially for large trees. In S-PLUS trees are made with the `tree` function and in R equivalent functionality is achieved with the `rpart` function. In this article we shall use the word tree but this can mean either `tree` or `rpart` depending on the package context. **pinktoe** contains tools that assist with the task of classifying or predicting the response value of a new case. For clarity of exposition we shall only describe classification although the tools work similarly for prediction of a response value in a regression tree.

Given a large tree object it is often difficult or impossible to visualise the whole tree or indeed other facets required for interpretation of the tree (e.g. associated text describing variables). Suppose one has a tree and wishes to classify a new case. The following two functions can help:

`PT` The `PT` function traverses the tree opening a Tcl/Tk GUI widget for each decision node in the tree and asking a user to make a decision based on the text in the widget. The decision is made by clicking a radiobutton and the results of this guide `PT` to use the

tree's rules to either open another decision widget or a leaf widget which conveys the results of the classification.

`pinktoe` The `pinktoe` function converts the tree to a set of web pages that enables a non-statistical user to make decisions using the tree in a clear and web-friendly manner. Although **pinktoe** itself does not produce glitzy web pages the pages it does produce are clear and simple.

The **pinktoe** package is not designed to produce a new kind of plot for tree objects. Other software and ideas could be proposed to improve the usability and classical display of trees but **pinktoe** does not do this. However, **pinktoe** is useful tool for the classification of new cases for large trees or those trees with complex variables. We now show how **pinktoe** can be used on a very simple example.

1.1. A simple example

Trees for classification and regression (CART) are one of the most popular and useful techniques for the analysis of multivariate data (see, for example, Breiman, Friedman, Olshen, and Stone (1984) or Clark and Pregibon (1993) for details on tree-based models). The usual presentation of a tree is through a *tree-diagram*. Both the popular S-PLUS and R packages provide functions to fit and assess tree models. They both provide various functions to plot trees graphically (see Clark and Pregibon (1993) or Venables and Ripley (1994)). To repeat the following example in R one needs to load the **rpart** package and load the `kyphosis` data set with

```
library("rpart")
data(kyphosis)
```

The command

```
z.kyphosis <- rpart(kyphosis)
```

builds a tree model using the well-known kyphosis data (see Chambers and Hastie (1993)). The constructed tree can be plotted and variable text added using the commands:

```
plot(z.kyphosis)
text(z.kyphosis)
```

The function `post.tree` produces the nice tree diagram shown in Figure 1:

```
post.tree(z.kyphosis)
```

The kyphosis data set is fairly simple: there are only 3 numerical variables `Start, Age` and `Number` and the response, `Kyphosis` is a factor with two levels `present` or `absent`. The tree plot in Figure 1 is extremely useful. The predicted kyphosis state for a new individual could easily be discovered by traversing the tree from root to leaf.

1.2. A more complex example: The need for pinktoe

In more complicated examples a simple tree diagram is not so effective. For example, Nason (2001) performed a contemporary statistical analysis of Early Day Motions (EDMs) from

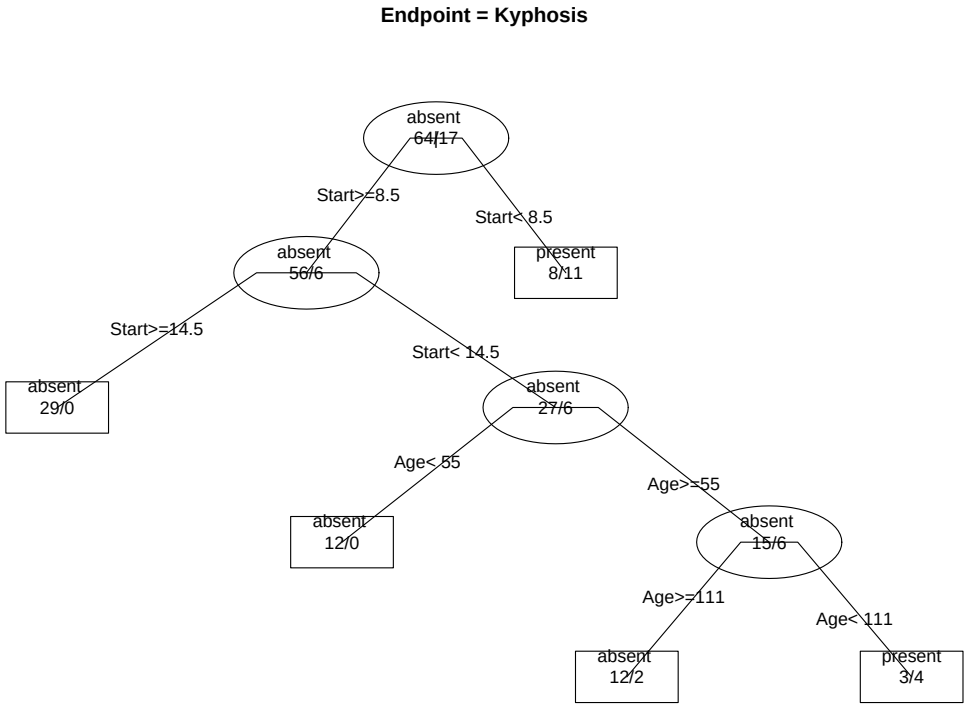

Figure 1: Kyphosis tree plot. To classify a new individual find a path down the tree from the root node at the top by evaluating each decision node (oval). For example, the first decision asks "Is Start < 8.5?" or not. An individual is classified when a rectangular leaf node is reached. Note that the variable and leaf text is very clear in this figure. There is even enough room for each node to present other information.

the three UK parliamentary sessions during 1997–2000. EDMs are expressions of opinion originating from backbench Members of Parliament (MPs). A motion can be initiated by any MP and then can be supported by any other MP (usually members of the executive, Ministers and the like, are not involved). EDMs are not expected to become law but they are a good way to generate publicity about an issue, or possibly embarrass the executive. More details on EDMs can be found in Berrington (1973), House of Commons Information Office (2000) or Nason (2001).

EDMs have been statistically analysed before. For example, see Finer, Berrington, and Bartholomew (1961), Berrington (1973), Franklin and Tappin (1977), Leece and Berrington (1977). However, two recent developments have encouraged the use of modern multivariate techniques for the analysis of EDM data. First, the annual number of EDMs per year has grown dramatically: the mean number of EDMs per year has increased from about 60 per year in the 1940s to about 1400 in the 1990s. Secondly, all EDMs since the 1997 General Election have been successively posted on the web at `edm.ais.co.uk`.

Figure 2 shows a 19-node decision tree computed using the following commands:

```
library("pinktoe")
data(mpincdf99)
z.edm <- makeEDMtree()
post(z.edm)
```

The multivariate data frame, `mpincdf99`, recorded which of 549 MPs (cases) signed any of the 1243 EDMs (variables). So, for example, the first row in the data frame, corresponding to Diane Abbott, MP., records that she signed EDM1, EDM3, EDM11, EDM17, EDM18, EDM19 and 245 further EDMs. In the tree construction the response variable for each case was the party political affiliation of each MP (e.g. Diane Abbott is Labour). The classification tree in Figure 2 was built to discover which types of EDM are related to party membership. Suppose one wanted to use the tree to predict the party affiliation of a new MP. Like the kyphosis tree earlier one could traverse the tree in Figure 2 from root to a leaf. However, there are two problems. First, the tree plot is clearly incomprehensible! Further, without extra information it is impossible to carry out a classification. For example, the first decision is whether or not the subject under consideration signs EDM142. What is EDM142? In fact, EDM142 is entitled *Government's Climate Change Policy* and its "variable description" is quite long. It reads

> "That this House congratulates the Government upon its manifesto commitment to reduce United Kingdom carbon dioxide emissions by 20 per cent. based on 1990 levels by 2010; further congratulates the Prime Minister for making it clear in the House that this is not a conditional target (Official Report, 24th June 1997, columns 687-8); fully supports the Deputy Prime Minister's statement at the Fifth Conference of the Parties to the Convention on Climate Change in Bonn in November in which he stressed the need for urgent domestic action to reduce greenhouse gas emissions, and that the Kyoto commitment to reduce greenhouse gas emissions by 12.5 per cent. between 2008 and 2012 is only a starting point, and emphasised his commitment to the domestic target of reducing carbon dioxide emissions by 20 per cent. by 2010; looks forward to the setting out and introduction of policies and measures to meet both the Kyoto and the 20 per cent. Manifesto targets in the

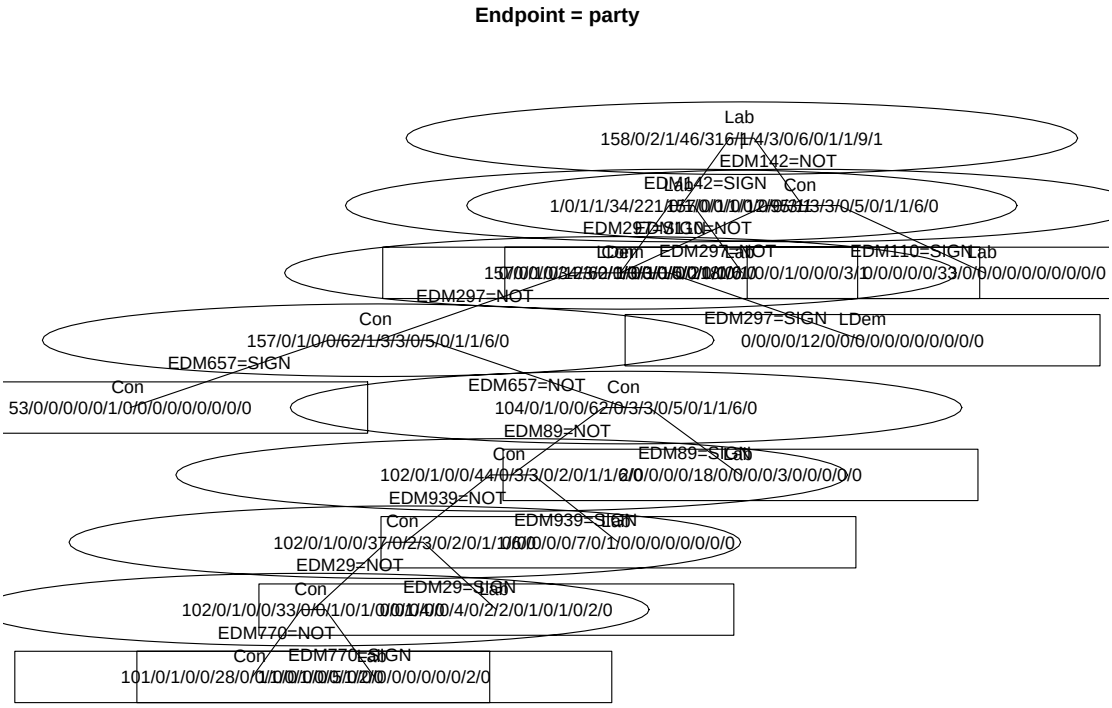

Figure 2: A 19 node tree plot for classifying MPs into political parties built from EDMs in the 1999–2000 session from Nason (2001). The plot was produced using the default `post.tree` function in R. Even though the full variable text is not displayed on this tree diagram the decision text is already overlapping and very unclear.

> Government's forthcoming Climate Change Strategy; and acknowledges that this will enable the Government to continue to lead the international battle against dangerous climate change."

This text can be reproduced using an example function in the **pinktoe** package via the commands:

```
data(edmbigtext)
edm.text("EDM142")
```

To decide whether an MP would sign EDM142 or not requires the resolution text to be read and understood. To traverse the tree one needs access to the decision tree *but also* the text of the EDMs. The EDM text could be added to the plot in Figure 2 but it is likely that the plot would be rendered even more incomprehensible.

Alternatively, two pieces of paper could be used: one with a comprehensible plot of the tree and another detailing the EDM text. However, some of the EDMs are longer than the one above and other good trees exist that are more complicated than that shown in Figure 2. The next section describes two solutions to the problem of traversal of trees which are large and/or have a large amount of "variable text".

2. pinktoe

2.1. pinktoe: Using widgets to traverse trees

To classify a new case in the EDM example one needs to traverse the tree from root to leaf and for each decision node one needs to know the variable text and then make a decision. The `PT` function does just this using Tcl/Tk widgets. For more details on Tcl/Tk widgets and their implementation in R see Dalgaard (2001, 2002).

Following on from the previous example the commands:

```
PT(z.edm, textfn=edm.text)
```

produce the widget as depicted in Figure 3. One can clearly read the variable text and then make a decision about whether or not the new MP would sign the EDM and click the submit button. If NOT is submitted for EDM142 then another widget is opened asking whether or not the MP would sign EDM297. If it was SIGNed then a leaf widget, like the one in Figure 4, is displayed which indicates that an MP which did not sign EDM142, but did sign EDM297 is predicted to be a Labour MP.

To run another example with the `kyphosis` data try:

```
PT(z.kyphosis)
```

and one can classify new cases and for this simpler example one does not require any extra text.

`PT` is extremely simple to use. One can merely supply one argument which is the tree that you wish to traverse. However, to utilize its full potential one can supply a second argument

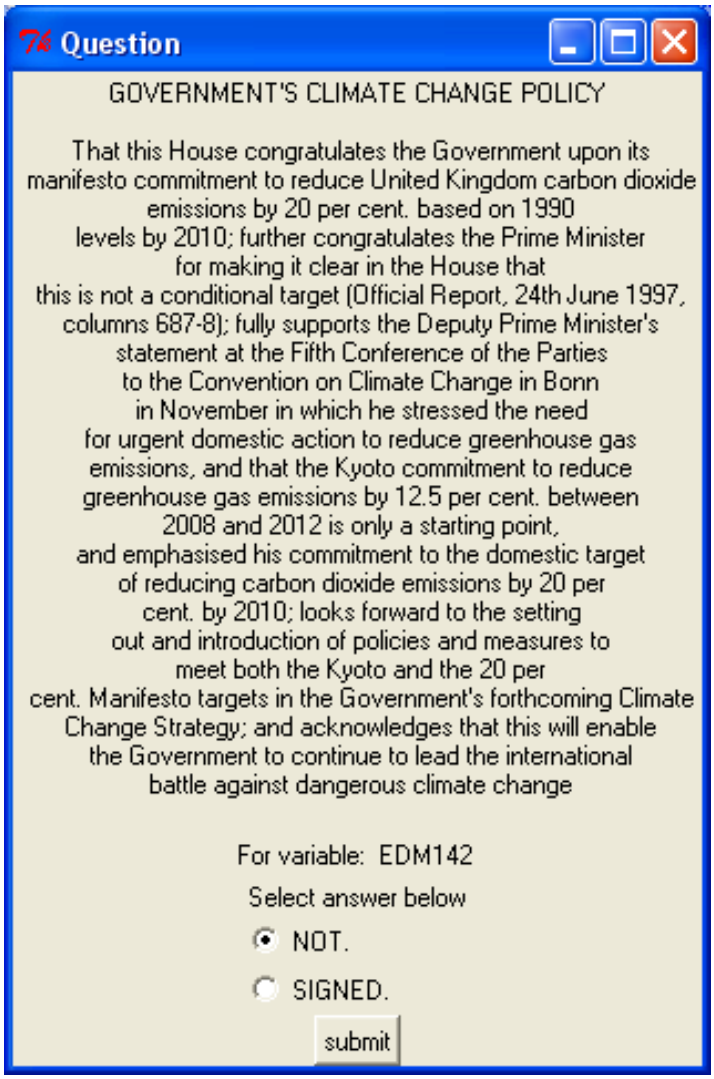

Figure 3: Tcl/Tk widget displaying the text of EDM142 and permitting the opportunity to agree or disagree (SIGN or NOT SIGN) the EDM. Once the decision has been made the appropriate radiobutton can be selected and the submit button pressed. The result gets sent back to PT.

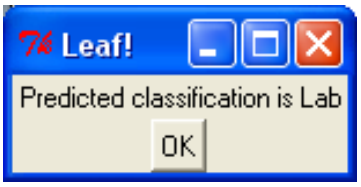

Figure 4: Leaf widget which is displayed when tree traversal ends.

`textfn`. The `textfn` argument is a user-supplied function that takes in a variable name and produces some text. In the EDM example the variable names are strings such as "EDM142", in the kyphosis example they are "Start", "Age" and "Number". The variable names involved in a `"tree"` or `"rpart"` object can be seen as the `var` component of the `frame` dataframe component of the tree (e.g. `z.edm$frame$var`). The variable names stem from the data frame that built the tree but not all variables make it into the tree. For example, the `edm.text` function call

```
edm.text("EDM142", web=FALSE)
```

produces the character string that contains the text for EDM142 displayed in Figure 3. The second argument to `textfn` should be `web` which when `FALSE` returns the text as a single character string (containing newlines, tabs, if necessary). If one intends to use the more sophisticated `pinktoe` function below then the `web=TRUE` argument should produce the text as HTML (which is what `edm.text` does).

Thus as `PT` traverses the tree and reaches a node it discovers which variable is involved in the decision and passes this variable name to `textfn` which returns the appropriate user-supplied text to the widget.

2.2. Traversing trees on the web

The `PT` algorithm described in the previous section is extremely useful if one wishes to semi-automatically traverse a tree especially where extra, comprehensive, text needs to be supplied to enable decisions to be made. However, use of `PT` requires that the user has R installed. It might be the case that users have a need to traverse a tree but have no desire or general need to have R installed on their machines (surprising as this may seem).

The `pinktoe` function is an elaboration of `PT` in that it can produce a set of HTML and Perl files that reproduce the behaviour of `PT` using a standard web-browser. Put simply the HTML files contain the decision text for a particular node and a check box. Submission of the check box calls an associated Perl file which works out whether another node's HTML is to be displayed or whether a leaf has been reached and to display some other kind of code.

Obviously R is required to create the HTML and Perl files using `pinktoe`. However, once created they can be installed on any CGI-enabled web server which also has Perl installed.

For example, Figure 5 shows the first web page from the tree sequence based on an 8-node tree which asks the first question about EDM297. To make the decision the user decides whether to check the box and then clicks on "Next" to either see the next decision or obtain the final classification. From EDM297 the next decision in the tree is either EDM142 if they did not select the checkbox or be told that they are a Liberal Democrat if they did.

The whole set of web pages for this example EDM tree can be traversed using the files created by **pinktoe** at http://www.stats.bris.ac.uk/~magpn/Research/Politics/TREE/best7/EDM297.htm. One can reproduce the files that make up this example by issuing the `pinktoe` command found in the Examples section of its R help file:

```
pinktoe(z.edm, edm.text, partytittext, treeid="",
    localdir="./",
    cgibindir="/~magpn/cgi-bin/TEST/",
```

EDM297: *INVESTMENT IN THE NHS*

That this House regrets that the Government has chosen to put tax cuts this April ahead of investment in the health service; and calls on the Government to re-direct the Âč2.7 billion of lost tax revenue into the NHS.

Check the box if you would sign EDM297 □
Next Reset

Return to *Who to Vote For* page

Figure 5: First decision to be made in the EDM 8 node tree. This is the actual first HTML page produced by **pinktoe** when displayed by a browser.

```
htmldir="/home/magpn/public_html/TEST/",
stateprintfn=partyprint,
requirelib="../party.lib",
commonhtml=partycommonhtml)
```

This function creates several `.html` and `.pl` files. One can peruse these and see that they call each other according to the decision nodes in the tree. The arguments are explained in some detail in the next section.

2.3. The pinktoe algorithm and some implementation details

To host an interactive tree your web server has to be able to execute Perl files through CGI. Enabling a web server to run Perl scripts is beyond the scope of this article but we refer the reader to Castro (1999) for further information. We also strongly recommend that you know (or find out) the location of your web server's error log to diagnose any problems should the web files not work. For correct operation your system's Perl must possess the "Cat.pm" module which is easily obtained from the CPAN (Comprehensive PERL Archive Network). However, once set-up any user can traverse the tree providing their browser can handle forms (recent versions of Mozilla, Netscape and Internet Explorer can do this).

To generate the web files a valid `"tree"` (or `"rpart"`) object is required and four or (optionally) five user-supplied functions described below. We give a generic description of each and also refer to examples that created the EDM tree. More examples are given in the **pinktoe** package help pages. The functions are:

`textfn` a *text function* that supplies the "variable text" as described above. For the EDM example this is a function that prints out the EDM text given an EDM number.

`tittext` a function producing *title text*. Given a variable label (EDM number) this function produces a character string describing the variable. For the EDM example such a function could return the title of the EDM.

`stateprintfn` this function controls what happens when a leaf of the tree is reached. The function takes a response or `yval` from the tree and takes an appropriate action. In the

EDM example the `partyprint` function takes a `yval` such as `"Lab"` or `"Con"` and calls Perl functions such as `&Labour;` or `&Conservative;` which are defined in the optional `requirelib` defined next. The Perl functions can contain code that prints out any kind of HTML: in the EDM example HTML is produced that displays the party logo and a link to the home page of the political party.

`requirelib` a character string pointing to an (optional) library of Perl functions, that may have been called by `stateprintfn`. If used, this library should sit where Perl can find it (possibly the `cgi-bin` directory).

`commonhtml` a function that prints out HTML which is appended to the end of each HTML file produced by **pinktoe**. In the EDM example the `partycommonhtml` function prints out an HTML link to the EDM analysis home page.

The basic algorithm behind pinktoe is simple but the code is a bit labyrinthine (as it comprises of R code writing HTML and Perl and then some of the Perl code itself produces HTML code when executed). For each decision node in the tree **pinktoe** produces one HTML file and one Perl file. The HTML file contains a title, variable text (using `tittext` and `textfn`) and an HTML `<FORM>` element which arranges for the associated Perl file to be executed when the `Next` button is pressed. The initial part of the Perl file consists of parsing commands obtained from Castro (1999) and then a decision is made to see whether the check box in the HTML file has been selected or not (e.g. `if (exists $isedms{'Age'})` for the kyphosis example). From the result of the decision either another HTML file is called or if a `<leaf>` is encountered `stateprintfn` is called to supply the necessary Perl code to deal with the `yval` at the leaf.

3. Conclusions

pinktoe is an R package that enables a semi-automatic traversal of a tree object either through Tcl/Tk widgets in R (using `PT`) or by creation of a suite of HTML/Perl pages to enable web-traversal with a suitably equipped web browser (using the `pinktoe` function).

pinktoe is useful when confronted with a large tree or variables that cannot be described concisely as a whole web page (arbitrary HTML) or text in a widget can be dedicated to each decision and leaf node in a tree. This is in contrast to the standard tree diagram which quickly becomes incomprehensible for large trees or impossible with verbose variable text.

pinktoe Version 2 can be obtained from CRAN at `http://www.R-project.org/`. The **pinktoe** package is named after a Pinktoe tarantula (*Avicularia avicularia*) a type of tree spider since our software puts trees on the web (Version 1 only contained the web interface).

Acknowledgements

Nason was partially supported by EPSRC Advanced Research Fellowship, AF1664. He would like to thank two referees, an Associate Editor and the Editor for suggesting dramatic improvements to both **pinktoe** and this paper.

References

Berrington H (1973). *Backbench Opinion in the House of Commons*. Pergamon Press, Oxford.

Breiman L, Friedman J, Olshen R, Stone C (1984). *Classification and Regression Trees*. Wadsworth, Belmont.

Castro E (1999). *Perl and CGI for the World Wide Web*. Peachpit Press, Berkeley.

Chambers J, Hastie T (1993). *Statistical Models in S*. Chapman and Hall, London.

Clark L, Pregibon D (1993). "Tree-based Models." In J Chambers, T Hastie (eds.), "Statistical Models in S," pp. 377–419. Chapman and Hall, London.

Dalgaard P (2001). "A Primer on the **R-Tcl/Tk** Package." *R News*, **1**(3), 27–31.

Dalgaard P (2002). "Changes to the **R-Tcl/Tk** Package." *R News*, **2**(3), 22–25.

Finer S, Berrington H, Bartholomew D (1961). *Backbench Opinion in the House of Commons 1955–59*. Pergamon Press, Oxford.

Franklin M, Tappin M (1977). "Early Day Motions as Unobstrusive Measures of Backbench Opinion in Britain." *British Journal of Political Science*, **7**, 49–69.

House of Commons Information Office (2000). "Early Day Motions Factsheet 30." ISSN 0144-4689.

Leece J, Berrington H (1977). "Measurements of Backbench Attitudes by Guttman Scaling of Early Day Motions: a Pilot Study, Labour 1968–69." *British Journal of Political Science*, **7**, 529–549.

Nason G (2001). "Early Day Motions: Exploring Backbench Opinion During 1997–2000." *Technical Report 01:11*, Statistics Group, Department of Mathematics, University of Bristol, Bristol, UK.

Venables W, Ripley B (1994). *Modern Applied Statistics with S-PLUS*. Springer-Verlag, New York.

Affiliation:

Guy Nason
Department of Mathematics
University of Bristol
Bristol BS8 1TW, United Kingdom
E-mail: g.p.nason@bristol.ac.uk
URL: http://www.stats.bris.ac.uk/~guy/

Journal of Statistical Software
May 2005, Volume 14, Issue 1.
http://www.jstatsoft.org/

Submitted: 2002-09-13
Accepted: 2005-04-19