

Package ‘pinktoe’

April 19, 2009

Title Graphically traverse a tree via GUI widgets or web based system

Version 2.0

Date 2004-09-08

Author G P Nason

Description Package can either (i) graphically traverse trees (rpart) objects. If a new case needs to be classified then the PT function uses a GUI to ask questions to determine how a new case satisfies the decisions in a classification tree and comes up with its classified value (also for regression trees) (ii) alternatively, the pinktoe function can produce HTML/PERL code which can be used with a CGI-enabled web server so that a tree can be traversed on a web site.

Maintainer <g.p.nason@bristol.ac.uk>

License GPL (>= 2)

URL <http://www.stats.bris.ac.uk/~magpn/Research/Pinktoe/pinktoe.htm>

Repository CRAN

Date/Publication 2006-02-08 12:07:34

R topics documented:

edm.text	2
edmbigtext	3
edmnum99	4
ednumswehave	5
genedmhtml	6
genericcommonhtml	7
kyphosis.text	8
kyphosisprint	9
kyphosistittext	10
makeEDMtree	11
mpall3	12
mpinc99	13

mpincdf99	14
partycommonhtml	15
partyprint	16
partyttext	17
pinktoe	18
PT	21
pv99tompall	23
splits.rpart	24

Index	26
--------------	-----------

edm.text	<i>Example of a textfn function for pinktoe/PT</i>
----------	--

Description

An example textfn text function for the EDM data set

Usage

```
edm.text(n, web = TRUE, file = "", append = FALSE)
```

Arguments

n	A EDM variable name, such as "EDM29"
web	If TRUE then the output is catted and appended to an HTML file. If FALSE the function returns a single character string with the desired text (including newlines, etc)
file	The name of the file to cat the text to (if web==TRUE)
append	Usually should be TRUE (I know the default is FALSE! Flip)

Details

This functions produces text (either in HTML format which gets catted to a file, or returned as a single character string) in response to an EDM number. The text describes the EDM. The actual text is stored in the character vector `edmbigttext`, the numbers of EDMs that this character vector contains is given in the object `ednumswehave`.

Value

If `web==TRUE` then nothing is returned. Otherwise, a single string containing the EDM text is produced.

Note

Not intended for general user use

Author(s)

Guy P Nason

References<http://www.stats.bris.ac.uk/~magpn/Research/Pinktoe/Welcome.html>**See Also**[pinktoe](#)**Examples**

```

#
# Which EDMs do we have text for
#
data(edmnumswehave)
data(edmbigtext)
edmnumswehave
#[1] 29 52 89 110 125 142 175 297 346 391 445 476 493 516 657 770 939
#
#
# Ok. Lets look at the text for EDM29. We'll set web=TRUE to get HTML output
#
edm.text("EDM29", web=TRUE)
#<EM>RACE RELATIONS </EM>
#<P>
#That this House accepts the fact that the
#United Kingdom is a multi-racial society; reasserts the
#right of all individuals to enjoy equal rights
#irrespective of their race, colour or creed; deplores
#the fact that racial discrimination continues to blight
#the lives of many British citizens, and many
#other ethnic minorities who live in the United
#Kingdom; and urges the Home Secretary to implement
#the recommendations by the Commission for Racial Equality
#and reform the Race Relations Act 1976. <BR>
#
# Do the same thing but return as a character string
#
edm.text("EDM29", web=FALSE)
#[1] " RACE RELATIONS \n\n That this House accepts the fact that the\n United Kingdom is a m

```

edmbigtext

Character vector containing all text necessary for the edm.text function

Description

The function `edm.text` uses this data set to extract text for various EDMs. The numbers of the EDMs stored in this vector are listed in the vector [ednumswehave](#)

Usage

```
data(edmbigtext)
```

Format

A character vector containing 1729 elements.

Source

Originally the UK Houses of Parliament, but either through Her Majesty's Stationary Office or [edm.ais.co.uk](#).

Examples

```
data(edmbigtext)
#
# Using edm.text
#
edm.text("EDM939")
#<EM>USE OF PARLIAMENTARY TIME </EM>
#<P>
#That this House welcomes the recommendation by the
#Modernisation Committee to timetable business so that the
#last stand alone votes are taken at 10.00
#p.m. thereby ending the practice of making legislation
#in the early hours of the morning; notes
#that backbenchers and opposition parties are to be
#given control over how the time for debating
#a bill is used; and believes that this
#will make for better scrutiny, better legislation and
#better government. <BR>
```

ednum99

Complete list of EDM numbers for 1999

Description

A complete list of EDM numbers for 1999. This is also the second dimnames component of [mpinc99](#).

Usage

```
data(ednum99)
```

Format

A vector of 1243 EDM numbers including amendments. The numbers are stored as character strings since some of them have letters in! E.g. "930A1" is an amendment of "930".

Source

Originally the UK Houses of Parliament, but either through Her Majesty's Stationary Office or `edm.ais.co.uk`.

Examples

```
data(edmnum99)
#
# Let's just have a look at a few
#
edmnum99[11:20]
# [1] "11"  "12"  "12A1" "13"  "14"  "15"  "16"  "17"  "18"  "19"
#
# Note that EDM12 has an amendment also, EDM12A1
#
```

`ednumswehave`

List of the EDM numbers that can be accessed by the `edm.text` function

Description

Only some of the EDMs listed in `edmnum99` are present in the `edmbigtext` vector which `edm.text` accesses. The ones that are present are listed in the `ednumswehave` vector.

Usage

```
data(ednumswehave)
```

Format

An integer vector of 17 elements. Each one is an EDM number that can be accessed through `edm.text`

Source

Originally the UK Houses of Parliament, but either through Her Majesty's Stationary Office or `edm.ais.co.uk`.

Examples

```
data(ednumswehave)
#
# Not really worth an example
#
```

 genedmhtml

Generate HTML and perl files to enable tree traversal

Description

Called by pinktoe to traverse a tree and generate appropriate perl and HTML files that enable a CGI-enabled web-server to graphically traverse a tree.

Usage

```
genedmhtml(treeobj, noderow, edmtextfn, edmtittext, cgibindir, htmldir, localdir, s
```

Arguments

treeobj	A tree or rpart object to traverse
noderow	The row to begin at
edmtextfn	A function that can print out explanatory text given a variable name (see web help page and <code>edm.text</code> example).
edmtittext	Prints out a title string given a variable name
cgibindir	A character string containing the directory where the perl files generated by pinktoe (with the extension <code>.pl</code>) will be stored. (This should be the directory part of the URL of the cgi-bin directory).
htmldir	A character string containing the directory where the HTML files generated by pinktoe (with the extension <code>.htm</code>) will be stored. (This should be a pathname understood and able to be found by perl).
localdir	A local location to store both the HTML and perl files immediately after they are generated
stateprintfn	A user-supplied function that decides what to do when supplied with the <code>yval</code> reached at the leaf of a tree. Some text can be output, or maybe a perl function call.
requirelib	A library of perl functions that can be called by, e.g. <code>stateprintfn</code> . The library that this refers to should reside in the <code>cgibin</code> directory. If no function calls are planned then it doesn't matter what argument is supplied.
commonhtml	A user-supplied function that prints out some HTML code. This is appended to every HTML web page.

Value

No deliberate return value.

Note

Not intended for general user use

Author(s)

Guy P Nason

References

<http://www.stats.bris.ac.uk/~magpn/Research/Pinktoe/Welcome.html>

See Also

[pinktoe](#)

Examples

```
#  
#Not intended for general user use  
#
```

`genericcommonhtml` *An example of some common html to go at the end of HTML pages for Pinktoe*

Description

Title says it all. This function could be supplied to [pinktoe](#) as the `commonhtml` argument.

Usage

```
genericcommonhtml(file, append)
```

Arguments

<code>file</code>	The name of the HTML file to append the HTML to
<code>append</code>	Should be TRUE

Details

Title says it all, see example below.

Value

No value returned.

Note

Not intended for general user use

Author(s)

Guy P Nason

References

<http://www.stats.bris.ac.uk/~magpn/Research/Pinktoe/Welcome.html>

See Also

[pinktoe](#)

Examples

```
#
# The "" makes the output go to standard output (ie on the console)
#
genericcommonhtml(file="", append=TRUE)
#<a href="http://www.stats.bris.ac.uk/~magpn">Return to <EM>Guy Nason's</EM> home page</a>
#
# Of course this is an example function which you can modify to say whatever you like!
```

kyphosis.text	<i>Print explanatory text concerning kyphosis variables.</i>
---------------	--

Description

Given a variable name this function prints out explanatory text in HTML

Usage

```
kyphosis.text(n, file = "", append = FALSE)
```

Arguments

n	A kyphosis variable. Could be "Start", "Age", "Number"
file	File to write HTML to
append	Should be TRUE

Details

As Description. This is an example function of a possible argument to the `textfn` argument of [pinktoe](#)

Value

No value is returned

Note

Not intended for general user use

Author(s)

Guy P Nason

References<http://www.stats.bris.ac.uk/~magpn/Research/Pinktoe/Welcome.html>**See Also**[pinktoe](#)**Examples**

```
#
# Here is a simple example
#
kyphosis.text("Start")
#<BR>
#Variable: Start : The beginning of the range of vertabrae involved<BR>
```

kyphosisprint	<i>Example stateprintfn for the pinktoe function arising from the kyphosis example</i>
---------------	--

Description

This function is an example of the type of function that could be supplied to the `stateprintfn` argument of the `pinktoe` function. Given a `yval` from `z.kyphosis` tree (such as "present" or "absent") it produces perl code to call an appropriate perl function (it actually calls the `present` or `absent` perl functions that are defined by the `requirelib` argument of `pinktoe` function

Usage

```
kyphosisprint(yval, file = "", append = FALSE)
```

Arguments

<code>yval</code>	A <code>z.kyphosis</code> <code>yval</code> . This is either "present" or "absent"
<code>file</code>	Where to cat the output
<code>append</code>	Should be TRUE

Details

As Description

Value

No value returned

Note

Not intended for general user use

Author(s)

Guy P Nason

References

<http://www.stats.bris.ac.uk/~magpn/Research/Pinktoe/Welcome.html>

See Also

[pinktoe](#)

Examples

```
#
# A quick example. The file here is specified as "" to cat to standard output
#
kyphosisprint("present")
# &present;
```

kyphosistittext	<i>An example of a tittext function for the pinktoe function with the kyphosis example.</i>
-----------------	---

Description

Returns a character string in response to a variable. Serves as an example tittext function for the [pinktoe](#) function.

Usage

```
kyphosistittext(label)
```

Arguments

label	A variable label. Could be "Start", "Age", "Number".
-------	--

Details

This is an example function that could be supplied as the tittext argument for the [pinktoe](#) function. All this function does is return a particular character string in response to a text string argument (which is supplied by the calling function as the variable from the tree in question). The returned character string is used as the title for the associated web page that gets produced for each HTML file produced by [pinktoe](#).

Value

A character string containing the response text.

Note

Not intended for general user use

Author(s)

Guy P Nason

References

<http://www.stats.bris.ac.uk/~magpn/Research/Pinktoe/Welcome.html>

See Also

[pinktoe](#)

Examples

```
#  
# Try it out with the variable name "Start"  
#  
kyphosistittext("Start")  
#[1] "Variable: Start"
```

makeEDMtree

Construct a tree for one of the examples showing how pinktoe works

Description

Merely constructs a tree ([rpart](#)) object.

Usage

```
makeEDMtree()
```

Details

Constructs a particular tree using a given data frame and a list of variables (the ones in [ednumswehave](#). When transferring code between S and R sometimes tree objects (in S) differ from their "equivalent" partners ([rpart](#) objects) in R. Hence it is useful to sometimes be able to recompute them for the particular package in question.

Value

An object of class `rpart` which is the made tree.

Note

Useful for constructing an example

Author(s)

Guy P Nason

References

<http://www.stats.bris.ac.uk/~magpn/Research/Pinktoe/Welcome.html>

See Also

[pinktoe](#)

Examples

```
#  
# Build the tree after loading rpart library  
#  
library("rpart")  
data("mpincdf99")  
mpincdf99tree <- makeEDMtree()
```

mpall3

Master database of MP information concerned with EDM example

Description

A list containing several components. Each component is some kind of vector with each row of each vector containing information about a given MP. Not really used in the examples but useful for the author to have around.

Usage

```
data(mpall3)
```

Format

The format is: List of 7 components \$ mpn : Factor w/ 670 levels containing the names of the MPs \$ mpcon : Factor w/ 660 levels containing the constituencies of MPs \$ mpparty : Factor w/ 16 levels containing MPs' party affiliation. \$ mpreregion: Factor w/ 9 levels containing the region containing the MPs' constituencies. \$ mpgender: Factor w/ 2 levels gender of MP. \$ mpage : num [1:670] Age of MP at given date. \$ mpagecod: num [1:670] Age range of MP

Source

Nason, G.P. (2001) Early Day Motions: exploring backbench opinion during 1997-2000. *Technical Report* 01:11, Department of Mathematics, University of Bristol.

Examples

```

data(mpall3)
#
# The name of MP number 3
#
mpall3$mpn[3]
#
# This record corresponds to Irene Adams
#
#[1] Adams, Irene
#670 Levels: Abbott, Miss Diane Adams, Gerry# Adams, Irene ... Young, Rt Hon Sir George Bt
mpall3$mpcon[3]
#
# Her constituency is Paisley North
#
#[1] Paisley North
#660 Levels: Aberavon Aberdeen Central Aberdeen North ... York, City of
mpall3$mpparty[3]
#
# She is a Labour MP
#
#[1] Lab
#16 Levels: Con DCWM DU Ind LDem Lab Lib PC SDLP SF SNP SPK Scot Lab UKU ... WW
mpall3$mpregion[3]
#
# Her constituency is in the North West of England
#
#[1] NorthWest
#9 Levels: London Midland NI NorthEast NorthWest Scotland ... Wales
#
# She is Female
#
mpall3$mpgender[3]
#[1] F
#Levels: F M
mpall3$mpage[3]
#
# She was 53 at the census date
#
#[1] 53
mpall3$mpagecod[3]
#
# She was in her fifties
#
#[1] 5

```

Description

A binary matrix containing a zero/one entry for each cell which represents whether or not a given MP signed a given Early Day Motion.

Usage

```
data(mpinc99)
```

Format

There are 549 MPs and 1243 EDMs.

Source

Nason, G.P. (2001) Early Day Motions: exploring backbench opinion during 1997-2000. *Technical Report 01:11*, Department of Mathematics, University of Bristol.

Examples

```
data(mpinc99)
## maybe str(mpinc99) ; plot(mpinc99) ...
#
# Did Irene Adams sign EDM3 ?
#
mpinc99[2, 3]
# [1] 0
#
# No, she didn't!
```

mpincdf99

EDM signing database for 1999 and party affiliation for MPs

Description

A dataframe. Each row represents an MP. The first entry in each row contains the party affiliation for that MP. Subsequent entries correspond to the factor SIGNED/NOT SIGNED indicating whether that MP did or did not sign the EDM described by that column.

Usage

```
data(mpincdf99)
```

Format

A dataframe containing 549 MPs (rows) and 1244 columns (one for party membership and the rest for EDMs).

Source

Nason, G.P. (2001) Early Day Motions: exploring backbench opinion during 1997-2000. *Technical Report* 01:11, Department of Mathematics, University of Bristol.

Examples

```
data(mpincdf99)
## maybe str(mpincdf99) ; plot(mpincdf99) ...
#
# Let's look at the first 5 MPs and the first 4 EDMs.
#
mpincdf99[1:5,1:5]
#
# party      EDM1 EDM2   EDM3   EDM4
#Abbott/Diane      Lab SIGNED  NOT SIGNED  NOT
#Adams/Irene       Lab   NOT   NOT   NOT SIGNED
#Ainger/Nick       Lab   NOT   NOT   NOT   NOT
#Ainsworth/Peter   Con   NOT   NOT   NOT   NOT
#Alexander/Douglas Lab   NOT   NOT   NOT   NOT
#
# So, Diane signed EDM1 and EDM3 and not EDM2 and EDM4 etc....
```

partycommonhtml *Function to produce HTML at the end of all EDM HTML pages*

Description

Often at the end of an HTML file code which is common across a set of pages is put. For example, a link back to some contents page. This function produces such code for the EDM example.

Usage

```
partycommonhtml(file, append)
```

Arguments

file	The HTML file to append the code to
append	Should always be TRUE

Details

Generates HTML code which is going to be common to all pages generated by a given [pinktoe](#) run.

Value

No value

Note

Not intended for general user use

Author(s)

Guy P Nason

References

<http://www.stats.bris.ac.uk/~magpn/Research/Pinktoe/Welcome.html>

See Also

[pinktoe](#)

Examples

```
#
# A simple example where filename="" to show result on the screen
#
partycommonhtml("", TRUE)
# <a href="http://www.stats.bris.ac.uk/~magpn/Research/Politics/whovote.htm">Return to <EM>W
```

partyprint

Function suitable for use as the stateprintfn argument for pinktoe for the EDM example

Description

In response to a variable (from the var component of the frame component of the relevant tree or [rpart](#) object) this function issues perl commands which call functions in the partylib object (which can be supplied as a `requirelib` object for [pinktoe](#))

Usage

```
partyprint(yval, file = "", append = FALSE)
```

Arguments

yval	One of "Lab", "Con", "LDem", "UU", "PC", "SNP": parties in the British Political system
file	perl file to write the output code to
append	Should be TRUE

Details

This file should produce perl code to do something when a yval is produced. A yval is one of the elements of the var column in the frame dataframe of a tree or `rpart` object. In this case the elements are all political parties indicated by the possibilities described in the `yval` argument to this function. Here perl code is generated which calls a function, one for each political party. See example below.

Value

Nothing.

Author(s)

Guy P Nason

References

<http://www.stats.bris.ac.uk/~magpn/Research/Pinktoe/Welcome.html>

See Also

[pinktoe](#)

Examples

```
partyprint("Lab", file="", append=TRUE)
# &labour;
#
#
# So the perl function labour gets called.
```

partyttext	<i>Returns the variable label</i>
------------	-----------------------------------

Description

For the EDM example the title text for each node is simply the EDM number which gets supplied as a character argument. So it just gets returned to be used as a title (for the eventual HTML page). Of course, one could modify this function to do something more complicated.

Usage

```
partyttext(v)
```

Arguments

v	A character string describing the variable under consideration, like "EDM29", etc
---	---

Details

Just returns the variable name which is good enough for a title in this situation.

Value

Returns its argument, the variable name.

Note

Not intended for general user use

Author(s)

Guy P Nason

References

<http://www.stats.bris.ac.uk/~magpn/Research/Pinktoe/Welcome.html>

See Also

[pinktoe](#)

Examples

```
#
# Try it with the "EDM29" variable name
#
partytittext("EDM29")
# [1] "EDM29"
```

pinktoe

Pinktoe: convert S trees to web files for interactive traversal.

Description

Pinktoe converts a S tree object to a set of HTML and perl files that can be, once uploaded to a perl-aware web server, interactively traversed by a user with a web browser. This is useful for large trees or trees where the variables require description by verbose text.

Usage

```
pinktoe(treeobj, textfn, tittext, treeid = "",
        cgibindir = paste("~/magpn/cgi-bin/", treeid, "/", sep = ""),
        htmldir = paste("/home/magpn/public_html/Research/Politics/TREE/",
        treeid, "/", sep = ""), localdir = "Tree/", stateprintfn = partyprint,
        requirelib = "../party.lib", commonhtml)
```

Arguments

treeobj	An object of class rpart (or tree in S)
textfn	A user supplied function that prints out text to a file in response to a variable name.
tittext	A user supplied function that prints out title text to a file in response to a variable name
treeid	This is a character string which is appended to both <code>cgibindir</code> and <code>htmlmdir</code> . This can be useful when you are building web pages for a collection of trees and store different trees in separate directories.
cgibindir	A character string containing the directory where the perl files generated by pinktoe (with the extension <code>.pl</code>) will be stored. (This should be the directory part of the URL of the cgi-bin directory).
htmlmdir	A character string containing the directory where the HTML files generated by pinktoe (with the extension <code>.htm</code>) will be stored. (This should be a pathname understood and able to be found by perl).
localdir	A local location to store both the HTML and perl files immediately after they are generated
stateprintfn	A user-supplied function that decides what to do when supplied with the <code>yval</code> reached at the leaf of a tree. Some text can be output, or maybe a perl function call.
requirelib	A library of perl functions that can be called by, e.g. <code>stateprintfn</code> . The library that this refers to should reside in the <code>cgibin</code> directory. If no function calls are planned then it doesn't matter what argument is supplied.
commonhtml	A user-supplied function that prints out some HTML code. This is appended to every HTML web page.

Details

See the example below for usage. See <http://www.stats.bris.ac.uk/~magpn/Research/Pinktoe/Welcome.html> for a full description

Value

No value is produced.

Note

This is version 2, an earlier version didn't work well with R

Author(s)

Guy P Nason

References

<http://www.stats.bris.ac.uk/~magpn/Research/Pinktoe/Welcome.html>

See Also[PT](#)**Examples**

```

#
# Attach rpart library
#
library("rpart")
data("mpincdf99")
#
# Create a tree (rpart object)
#
z.edm <- makeEDMtree()
#
# Plot the tree to see its basic structure.
#
plot(z.edm)
text(z.edm)
#
# Now use pinktoe to generate a set of htm and pl files that can be used
# by a CGI-enabled web server to traverse the tree.
#
data("edmbigtext")
#
# Next code requires "sfsmisc" library. This is not installed by default
# in R distributions so you'll have to get it yourself. We make sure
# that the next code doesn't run because R CMD check can't install packages (I think)
#
## Not run:
pinktoe(z.edm, edm.text, partytittext, treeid="", localdir=".",
        cgibindir="/~magpn/cgi-bin/TEST/",
        htmdir="/home/magpn/public_html/TEST/", stateprintfn=partyprint,
        requirelib="./party.lib",
        commonhtml=partycommonhtml)
## End(Not run)
# Frame row number is 1
# Node number is 1
# Frame row number is 2
# Node number is 2
# Frame row number is 3
# Node number is 4
# Frame row number is 4
# Node number is 8
# Frame row number is 6
# Node number is 17
# Frame row number is 7
# Node number is 34
# Frame row number is 8
# Node number is 68
# Frame row number is 9
# Node number is 136

```

```
# Frame row number is 17
# Node number is 3
#
# If you look in the current directory you'll find a load of perl and
# HTML files created.
#
```

PT

Traverse rpart object using Tcl Tk GUI

Description

Function uses Tcl/Tk widgets to enable a tree (or, more generally, an rpart object) to be traversed graphically. This is useful for complicated trees and/or where variable descriptions are long.

Usage

```
PT(rpart, textfn = NULL)
```

Arguments

rpart	An rpart object
textfn	A function that converts a variable name into text

Details

Suppose one has a rpart (or tree in Splus) classification or regression tree. A common task is to take a new case or individual and use the tree to make a decision about the new case (or to put it another way to classify the class membership of the new case).

One could use the tree plot to decide this and at each node make a decision and follow the edges of the tree to a leaf which contains the final classification value. However, with some trees there are two problems that PT is designed to overcome. The first is that some trees are huge. Plots of such trees are often incomprehensible. The second is that a description of the decision to be made at a node (or even every node) might require a significant amount of supportive text. It might be impractical to augment a tree plot with a large amount of text at each node as this would, again, render the tree plot incomprehensible.

This routine presents, for each node, starting with the root, a little window (widget) which asks the user to take their new case and answer a question in relation to the new case. The software takes their answer and automatically works out which is the next node to visit and question to ask. Eventually, it reaches a leaf and displays the predicated classification (or predicted value, for regression) for the new case (and it also returns it to the calling function).

The questions are answered by clicking the relevant radiobutton and then clicking submit.

A feature of PT is that for each question asked an optional amount of supportive text can be supplied. This text appears in the window where the question is being asked. This feature is useful as it means that the full information about the question being asked can be displayed altogether without, say, having to make a separate and inconvenient reference to a separate document.

The optional text is supplied through the `textfn` argument. This function should have one argument which can be any of the variable names in the tree (which, of course, are some of the original variables in the dataframe from which the tree was constructed. The appropriate text should be supplied as a single character string (although it can contain newline characters, etc). The `textfn` function should also have the argument `web` such that when this argument is `FALSE` a single string of text is produced (for this function). [When the `web` argument is `TRUE` then this is designed for use with the `pinktoe` function and it would be desirable for `textfn` to produce the same text but printed out (to a specified file) in HTML format.]

Value

The classification (or predicted value) for the new case whose characteristics that one specifies through answering the questions.

Note

This package contains another function `pinktoe` which produces a similar system but in terms of a CGI-HTML web based system (which is harder to use but produces code that can be accessed by any web browser by any one)

Author(s)

Guy P Nason

References

www.stats.bris.ac.uk/~magpn/Research/Pinktoe/pinktoe.htm

See Also

[pinktoe](#)

Examples

```
#
# Load rpart library (needed for tree building and execution of PT
#
library("rpart")
library("tcltk")
data("mpinCDF99")
#
# Use supplied function to create a suitable example.
#
z.edm <- makeEDMtree()
#
# Plot tree to look at the basic structure
#
plot(z.edm)
text(z.edm)
#
# Now use PT to graphically traverse tree
```

```

#
data("edmbigtext")
## Not run: PT(z.edm, textfn=edm.text)
#
# Successive widgets appear. You can answer each one using the
# default options if you like (the final classification will be
# "Conservative".
#
# If you have more time then notice that the order of questioning
# and your answers follow the edge structure of the tree.
#
# Notice the large amount of text supplied with each question (variable)
# This is achieved by the edm.text function.
#

```

pv99tompall

Permutation vector from MP lists in 99 into the mpall3 database

Description

In 1999 a selection of MPs signed Early Day Motions and hence turn up in databases that record which MPs signed in a given year. The lists that record MPs signing records implicitly define an "index position" for those MPs in those records. To link these MPs to the global [mpall3](#) database this vector gives the index into [mpall3](#) from their implicit index position.

Usage

```
data(pv99tompall)
```

Format

A permutation vector containing 549 integers each one identifying an MP and their position within the [mpall3](#) database.

Source

Nason, G.P. (2001) Early Day Motions: exploring backbench opinion during 1997-2000. *Technical Report 01:11*, Department of Mathematics, University of Bristol.

Examples

```

data(pv99tompall)
data(mpall3)
data(mpinc99)
#
# In the mpincdf object (which records EDM signings) Irene Adams is MP number 2. Let's check
#
dimnames(mpinc99)[[1]][1:5]
#[1] "Abbott/Diane"      "Adams/Irene"      "Ainger/Nick"

```

```

#[4] "Ainsworth/Peter"    "Alexander/Douglas"
#
# What number is she in the mpall3 database?
#
pv99tompall[2]
# [1] 3
#
# She is the third MP in this structure. (You see possibly number 1 or 2 in mpall3 did not s
# in 1999.
#
# Let's check
#
mpall3$mpn[3]
#[1] Adams, Irene
#670 Levels: Abbott, Miss Diane Adams, Gerry# Adams, Irene ... Young, Rt Hon Sir George Bt
#
# Yep, that's her!

```

splits.rpart

Create splits data frame that seems to be missing from R rpart code

Description

The tree function in S returns a tree class object which has frame as one of its components. This frame has two columns called splits.cutleft and splits.cutright which illustrate how a variable split was achieved for a given node. The R rpart function documentation says that it does produce these splits component but I have been unable to find it!! This function recreates, as best I can, the splits components which pinktoe makes use of.

Usage

```
splits.rpart(tree, print.it = FALSE)
```

Arguments

tree	The rpart object that you wish to produce splits info for
print.it	If TRUE then informative messages are printed out (useful for debugging)

Details

Produces the (missing) splits components for the frame component of an rpart object.

Value

Returns a data.frame with two columns and the same number of rows as the frame component of the input tree object. The first (second) column contains information about the left hand (right hand) edge from a node.

Note

Of course, the splits component could be lurking somewhere else in the rpart object but I have been unable to find it!

Author(s)

Guy P Nason

References

None.

See Also

[pinktoe, PT](#)

Examples

```

library("rpart")
data(kyphosis)
z.kyphosis <- rpart(kyphosis)
#
# Now check that there is no splits.cutleft or splits.cutright component
# of the frame component as there is in S
#
dimnames(z.kyphosis$frame)[[2]]
#
#[1] "var"          "n"            "wt"          "dev"         "yval"
#[6] "complexity" "ncompete"    "nsurrogate" "yval2"
#
# Nope! So let's use our function to make them
#
splits.rpart(z.kyphosis)
#
# splits.cutleft splits.cutright
#1          >=8.5          < 8.5
#2          >=14.5         < 14.5
#3
#4          < 55           >=55
#5
#6          >=111         < 111
#7
#8
#9
#
# The resultant object could be installed into the frame component
# (as happens in the early code lines of PT and pinktoe) by
#
# z.kyphosis$frame <- cbind(z.kyphosis$frame, splits.rpart(z.kyphosis))

```

Index

*Topic **datagen**

edm.text, 1
genedmhtml, 5
kyphosis.text, 8
kyphosisprint, 9
kyphosistittext, 10
makeEDMtree, 11
partycommonhtml, 15
partyprint, 16
partytittext, 17

*Topic **datasets**

edmbigtext, 3
edmnum99, 4
ednumswehave, 5
mpall3, 12
mpinc99, 13
mpincdf99, 14
pv99tompall, 23

*Topic **iplot**

pinktoe, 18
PT, 20
splits.rpart, 24

*Topic **misc**

genericcommonhtml, 7

*Topic **tree**

pinktoe, 18
PT, 20
splits.rpart, 24

edm.text, 1
edmbigtext, 3
edmnum99, 4
ednumswehave, 3, 5, 11

genedmhtml, 5
genericcommonhtml, 7

kyphosis.text, 8
kyphosisprint, 9
kyphosistittext, 10

makeEDMtree, 11

mpall3, 12, 23

mpinc99, 4, 13

mpincdf99, 14

partycommonhtml, 15

partyprint, 16

partytittext, 17

pinktoe, 2, 6–12, 15–17, 18, 18, 22, 25

PT, 19, 20, 25

pv99tompall, 23

rpart, 11, 16

splits.rpart, 24