Package 'sdtoolkit'

February 20, 2015

Type Package

Title Scenario Discovery Tools to Support Robust Decision Making

Version 2.33-1

Date 2014-02-16

Author Benjamin P. Bryant

Maintainer Benjamin P. Bryant <bpbryant@gmail.com>

Description Implements algorithms to help with scenario discovery - currently only modified version of the the Patient Rule Induction Method.

License GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2014-02-16 20:39:43

R topics documented:

`sdtoolkit-package` *Scenario Discovery Tools to Support Robust Decision Making*

Description

This package is designed to provide a means for easily integrating multiple algorithms for scenario discovery and assessment, an element of the Robust Decision Making process for decisionmaking under uncertainty. It currently implements a coverage-oriented version of Friedman and Fisher's Patient Rule Induction Method (PRIM), and provides additional diagnostic tools to assess the quality of the scenarios that emerge.

Details

Package:	sdtoolkit
Type:	Package
Version:	2.32-3
Date:	2011-10-10
License:	GPL3
Code Copyright:	Evolving Logic, Inc.
Documentation Copyright:	The RAND Corporation
LazyLoad:	yes

This package provides interactive functions for scenario discovery that should be usable by those not particularly fluent in R, as well as more direct and powerful code that can be used or extended by those more familiar with both R and the scenario discovery process. The package is built around the function `sdprim` ("Scenario Discovery PRIM"), which implements a very interactive, diagnostic-laden, slightly modified version of Friedman and Fisher's Patient Rule Induction Method. The function `sd.start` is a helper function that aids the user in reading in their data, checking for and resolving inappropriate data features, and exploring different output variables and potential thresholds.

In addition, there are several *post-discovery* functions that allow the user to visualize the results of running the algorithm(s). The are currently three high-level functions of interest: `seq.info`, which redisplays all information identified by the algorithm, including the definition of all relevant boxes, and statistics for the entire sequence. The plotting function `dimplot` graphically displays normalized dimension restrictions defining a given box, and the plotting function `scatterbox` displays boxes over a scatterplot of the dataset, in which the points are color-coded according to their 0/1 value.

Lastly, there are many "almost external" functions for which the time constraints on cleaning up and documenting as user-level functions was slightly too high for this round of work. One line descriptions of all code functions can be found in the `undocumented` help file, which should be useful guidance for anyone looking to capitalize on existing but less thoroughly documented work.

Author(s)

Benjamin P. Bryant Maintainer: The same <bryant@prgs.edu>

Note of Acknowledgment: The fundamental functions used in the operation of PRIM (specifically, peeling and pasting to form a trajectory of induced boxes) were initially taken from Duong's **prim** package. Several pieces of code follow very closely the original functions provided by Duong, however because most functions used required small to large modifications to suit the specific needs of our scenario discovery task, it was easier to integrate these into **sdtoolkit** rather than build a package dependency on **prim**. The internal functions `peel.one` and `paste.one` were taken almost as is, and the primary workhorse function `find.traj` was derived heavily from Duong's `find.box`.

The funding for the development of this package came from Evolving Logic, RAND, and a National Science Foundation grant (SES-0345925) to the RAND Corporation. Evolving Logic provided funds for the majority of the customized algorithm implementation, the RAND Pardee Center for Longer Range Global Policy and the Future Human Condition supported the documentation of the package, and the National Science Foundation supported the research that led to the development of the scenario discovery methods employed in this package.

References

Bryant, B.P. and R.J. Lempert. (2010). "Thinking Inside the Box: A Participatory, Computer-Assisted Approach to Scenario Discovery". *Technological Forecasting and Social Change* 77: 34-49

Lempert, R.J., B.P. Bryant and S.C. Bankes. (2008). "Comparing Algorithms for Scenario Discovery". *RAND Working Paper Series*. http://www.rand.org/pub/working_papers/2008/RAND_WR557.pdf

Friedman, J.H., N.I. Fisher. (1999). "Bump hunting in high-dimensional data". *Statistics and Computing* 9, 123-143.

Groves, D.G. and R.J. Lempert. (2007). "A New Analytic Method for Finding Policy-Relevant Scenarios". *Global Environmental Change*, 17, p73-85. http://www.rand.org/pubs/reprints/RP1244/

Lempert, R.J., D.G. Groves, et al. (2006). "A general, analytic method for generating robust strategies and narrative scenarios". *Management Science* 52(4): 514-528.

Examples

```
## Not run:
#Note that all the examples shown here do not illustrate use of the extra
#options that can be passed to the functions as additional arguments.  The
#Individual function help files illustrate the use of optional
#arguments in more details.

#Also, here and in other example sections of code you may see the code preceded
#by "## Not run:" - this refers to sections of code being exempt from R's
#automatic example code checking due to their interactive nature.  Lines of code
#that immediately follow "## Not run:" should still be runnable, provided you
#remove the "## Not run:" comment.

### ===== SD.START: =====

#Users not familiar with importing and manipulating data in R will wish to start
#with:

\dontrun{
```

```
myboxes <- sd.start()
}

#This will prompt for things like directory and file name, and then walk through
#data inspection, thresholding, and offer to call sdprim.


### ===== SDPRIM: =====

#Users confident in the soundness and appropriate formatting of their data may
#take the following more direct actions:

\dontrun{

#LOAD the data, either via:
mydata <- read.csv("mycsvfile.csv")

#OR
loadedname <- load("myrdafile.rda")
mydata <- get(loadedname)

#Then define their input variables:
xmatrix <- mydata[,columnindexes]

#Then define their output variable using EITHER
outputvar <- mydata[,outputcolumnnumber]

#OR
outputvar <- mydata[,"outputname"]

#If output var is already a 0-1 variable, then sdprim can be called as:
myboxes <- sdprim(x=xmatrix, y=outputvar)

#Otherwise, first threshold the output variable as follows:
outthresh <- 1*(outputvar>threshold)

#Then call sdprim:
myboxes <- sdprim(x=xmatrix, y=outthresh)
}

### ===== SEQ.INFO: =====

#To see a summary of sdprim output,
data(exboxes)  #an example box sequence object included with the package
boxinfo <- seq.info(exboxes)


### ===== DIMPLOT: =====

#To see a 'Normalized Dimension Restriction Plot' for box i, type:
data(exboxes)
dimplot(exboxes, 1)
```

```
## End(Not run)
```

`dimplot` *Plot normalized dimension restrictions as horizontal bars*

Description

This function takes the output of `sdprim` and displays the dimension restrictions for a given box, normalized by the range of the data along each dimension.

Usage

```
dimplot(boxseq, boxind=1, alldims = FALSE, thetitle = NULL,
                incol = "lightblue", longcol = "black")
```

Arguments

`boxseq`	A box sequence object, as output by `sdprim`.
`boxind`	An integer, indicating which box in the box sequence to plot.
`alldims`	Logical, whether or not to plot all dimensions, or just those that have some dimension restrictions.
`thetitle`	Character, title to put on the plot. If none supplied, will default to "Normalized Dimension Restrictions".
`incol`	What color to use for the portion of the barplot representing the inside-the-box range. Must be one of those in `colors()`.
`longcol`	What color to use for the portion of the barplot covering values outside the box, extending from zero to unity. Must be one of those in `colors()`.

Value

Nothing returned, used for its plotting capability.

Author(s)

Benjamin P. Bryant, <bryant@prgs.edu>

See Also

See `sdprim` for the primary algorithm, and `seqinfo` for other display options.

Examples

```
#Load an example box object
data(exboxes)

#Plot the first box with only the restricted dimensions shown:
dimplot(exboxes, 1)

#Plot the second box showing all the dimensions, whether or not
#they are restricted:
dimplot(exboxes, 2, alldims=TRUE)
```

---

exboxes *Example Box Sequence Object*

Description

An example box sequence for demonstrating **sdtoolkit** functions that apply to box sequence objects. As of now these other functions are `seqinfo` and `dimplot`.

Usage

```
data(exboxes)
```

Format

The format is that of a box sequence object as output by `sdprim`. Please the 'Value' section of the documentation for `sdprim` for additional details.

Source

The object was generated by running `sdprim` on the common dataset `quakes`, with the fourth varable (`mag`) used as the output variable, thresholded at 5.0.

Examples

```
data(exboxes)
```

scatterbox *Plot scatterplots of data with boxes shown*

Description

Function for plotting scatterplots of cases, with points color-coded by relevance and with two-dimensional projections of scenarios-as-boxes.

Usage

```
scatterbox(boxseq, boxnum = 1, xdim, ydim, filterdims = FALSE,
  filterothers = FALSE, nobox = FALSE, addbox = FALSE,
  data = attr(boxseq, "data"), y = NULL, manualbox = FALSE, ...)
```

Arguments

boxseq	A box sequence object as ouput by sdprim. Alternatively, a manually created list of box definitions (see details.
boxnum	Integer - Which box in the box sequence should be plotted?
xdim	Integer or Character - Which dimension should be plotted as the x-axis? The safest way is to specify the dimension name. See details.
ydim	Integer or Character - Which dimension should be plotted as the y-axis? The safest way is to specify the dimension name. See details.
filterdims	Logical - If a box is defined by more than just the two dimensions that are being plotted, should points filtered out by the other dimensions be removed from the scatter plot? See details.
filterothers	Specifies whether and how points captured by other boxes should be filtered out. Either TRUE, FALSE, "PREV" or a vector of numerics. TRUE removes points captured by all other boxes. FALSE removes none. "PREV" removes any boxes that came earlier in the covering sequence. Alternatively, a vector of integers may be specified indicating which boxes should have their points removed. Note that in all cases, boxes may overlap and specifying this option may result in not all points in the current box being shown.
nobox	Logical - should the scatterplot be generated, but with no box plotted?
addbox	Logical - should a different box be plotted on the same scatterplot? Note that depending on how points were filtered and which dimensions are involved, this may or may not be misleading.
data	Matrix - the data to use in generating the scatterplots. The default is to use the data attribute returned by sdprim. If this is not used for some reason (such as training versus test data), the dataset should have the same number of dimensions, same dimension names, and same dimension ordering as that used to generate the boxes.
y	Vector of zeros and ones indicating interestingness or uninterestingness of points. If left as NULL, it will assume the item data above is of the form returned by sdprim.

manualbox	Logical - Is the boxseq argument given a manually generated box list not conforming to the output of sdprim? This may be handy if you want to plot boxes or box sequences that you specified yourself independent of sdprim. See details for how to construct such a box.
...	Arguments that may be passed on to pbox and colptplot.

Value

Nothing returned, used for its plotting capabilities.

Author(s)

Benjamin P. Bryant, <bryant@prgs.edu>

References

RAND Pardee Center: http://www.rand.org/international_programs/pardee/

Examples

```
data(exboxes)
scatterbox(exboxes,xdim="stations",ydim="depth")
```

---

sd.start *Read, Clean, And Threshold Data To Call PRIM*

Description

This function is designed to help make sure that data used in PRIM analysis is in an appropriate form. It checks for duplicate columns and columns with no changing values, and also for various forms of missing or non-real-valued data. It then displays histograms and empirical cdfs for candidate output variables, and lets you define thresholds, and then call sdprim.

Usage

```
sd.start(...)
```

Arguments

...	The function is interactive and prompts for inputs. However, you may prespecify arguments to pass on to sdprim when it is called. See that function's help file for much more detailed information.

Details

The current version of sdprim has some notable restrictions on the format of data it can accept while still functioning appropriately. In particular, the data must be numeric, and the output vector that is eventually used must be zero one. This function (sd.start) helps to make sure the input and output data is appropriately formatted prior to calling sdprim.

Value

If one chooses to call sdprim from within sd.start, then it will return the same output of sdprim. Specifically, a convoluted list containing all relevant box sequence information, with structure as described more precisely in the help file for sdprim.

If sdprim is not called, then nothing is returned (or possibly some junk), though you may save data files interactively from within sd.start.

Author(s)

Benjamin P. Bryant, <bryant@prgs.edu>

References

Lempert, R.J., B.P. Bryant and S.C. Bankes. (2008). "Comparing Algorithms for Scenario Discovery." *RAND Working Paper Series*. http://www.rand.org/pub/working_papers/2008/RAND_WR557.pdf

Friedman, J.H., N.I. Fisher. (1999). "Bump hunting in high-dimensional data." *Statistics and Computing* 9, 123-143.

Groves, D.G. and R.J. Lempert. (2007). "A New Analytic Method for Finding Policy-Relevant Scenarios". *Global Environmental Change*, 17, p73-85. http://www.rand.org/pubs/reprints/RP1244/

Lempert, R.J., D.G. Groves, et al. (2006). "A general, analytic method for generating robust strategies and narrative scenarios." *Management Science* 52(4): 514-528.

See Also

sdprim

Examples

```
#Note, because the function is interactive, this example demonstrates very
#little.  You will need a csv or rda file available to load data in.
#See the sdprim examples for more explicit use.

## Not run: myboxes <- sd.start()
#Uses all of sdprim's default arguments

## Not run: morebox <- sd.start(repro=FALSE)
#tells sdprim not to spend computatioal time on
#reproducibility statistics
```

sdprim *Patient Rule Induction Method Adapted For Scenario Discovery*

Description

This is the primary function for the sdtoolkit package. It organizes many subsidiary functions into an interactive session to aid the user in identifying policy-relevant scenarios. It is based on Friedman and Fisher's PRIM, but includes additional modifications and diagnostics to better suit the scenario discovery task.

Usage

```
sdprim(x, y = NULL,
    thresh = NULL,
    peel.alpha = 0.1,
    paste.alpha = 0.05,
    mass.min = 0.001,
    pasting = TRUE,
    box.init = NULL,
    coverage = TRUE,
    outfile = "boxsum.txt",
    csvfile = "primboxes.csv",
    repro = TRUE,
    nbump = 10,
    dfrac = 0.5,
    threshtype = ">",
    trajplot_xlim = c(0,1),
    trajplot_ylim = c(0,1),
    peel_crit = 1)
```

Arguments

x	Matrix of input variables - all values must be numeric at present. (No non-binary categorical values allowed.)
y	Output vector, which may or may not be thresholded to be zeros and ones. If not thresholded, you must specify both thresh and threshtype below.
thresh	Numeric - if y is not a zero-one vector, you must specify a real value on which to threshold the data.
peel.alpha	Peeling parameter for PRIM. Typically around .05 to .1.
paste.alpha	Pasting parameter for PRIM.
mass.min	Minimum fraction of original points that must remain to allow continued peeling.
pasting	Logical indicating whether or not to paste.

box.init	Matrix, providing specification of the initial box bounds, if for some reason you wanted to limit the area over which it searched. The format is 2 rows by ncol(x), with the first row specifying lower bounds on each dimension, and the second row specifying upper bounds. The matrix should have column names matching colnames(x).
coverage	Logical, indicating whether or not to provide coverage-oriented statistics during plotting, rather than support-oriented statistics
outfile	Optional character, naming a text file where the box summary information will be copied. Set equal to NA (no quotes) if you would not like a file written out.
csvfile	Optional character, naming a default csv file where CARs-readable output will be sent. Regardless of what you put here, sdprim will also ask if you would like to write it out as csv, and give you the chance to change the filename.
repro	Logical, specifying whether or not to automatically generate reproducibility statistics by rerunning PRIM on random samplings of the dataset. The only reason to set to FALSE is if there is a prohibitively long run time. Another alternative is to lower the number of resamplings, nbump, below.
nbump	Integer - If repro = TRUE, how many resamplings should be performed? Currently this only allows sampling without replacement.
dfrac	Numeric between 0 and 1. If repro = TRUE, what fraction of the dataset should be resampled each time?
threshtype	Required only if the output data is not already in a zero-one formate. A relational operator entered as a character (either "<",">","<=" or ">=", describing how to values in y should be thresholded. The threshold is treated as being on the right side of the operator, thus, for example, "<" would make all values in y that are less than than thresh ones, and all values greater into zeros.
trajplot_xlim	x-limits on the peeling trajectory plot. Must be either a vector that can be passed as xlim argument to plot function, or NULL to make x limits narrowly around the data.
trajplot_ylim	y-limits on the peeling trajectory plot. Must be either a vector that can be passed as ylim argument to plot function, or NULL to make y limits narrowly around the data.
peel_crit	Either 1, 2, or 3, 1 is default – choose peels based on maximizing density in remaining box. 2 is normalize improvement by peeled box support, 3 is minimize mean in peeled box. See F&F 1999 Sec 14.1.

Details

Here we discuss several terminology issues that will be useful in understanding the output below, and then describe the interactive process used to identify scenarios.

This package is generally oriented around producing *boxes*, which are defined by a set of orthogonal restrictions on the input (uncertainty) space of a model. In a policy context, a box or set of related boxes can be interpreted as a *scenario*. See the references at the end of this entry for a more thorough explanation of the concept of analytically generated scenarios.

Two important measures of scenario adequacy are termed *coverage* and *density*. These are defined based on the binary output variable, which typically denotes some measure of "interestingness" of

the input-output combination. Coverage is the ratio of interesting points (those with an output value of one) captured to the total interesting points in the dataset, and density is the fraction of captured points that are actually interesting (ones in the box to total points in the box). (These have analogues to Type I and Type II errors and other measures from information theory.)

The basic PRIM algorithm operates in two steps, first generating a *trajectory* of boxes (in coverage-density space) that provide a tradeoff frontier for scenarios, with coverage, density and the number of restricted dimensions generally in tension. From this trajectory, the user selects and further inspects one or more candidate boxes that appropriately balance the measures of interest. After identifying and possibly modifying a box, the data points contained within that box are removed from the input matrix, and a new trajectory is generated. This process (*covering*) can continue until the user is satisfied with the total coverage achieved, or until other conditions prohibit further covering. The resulting set of *selected* boxes is referred to as a *box sequence*, and is the primary product of the PRIM algorithm.

The `sdprim` algorithm is very interactive, and the user will receive several prompts while running it. Note that during this process, at least on MS Windows versions, you will receive a "busy" hourglass even when you have an activated R console awaiting user input. Input can still (must) be entered in this state.

The first asks whether they would like the peeling trajectory displayed with dimension contours, with dominating points, or in the standard form.

Dimension contours represent coverage and density values achieved by boxes while holding the total number of restricted dimensions constant. They do not represent a complete or optimal contour derived by considering the frontier of all possible boxes of a given dimension, but rather simply display the coverage and density associated with boxes that could be created by dropping dimensions (in order of least importance) from other boxes in the current trajectory. That is, if there is a 3 dimensional box with restrictions x1 < 1, x8 > 5, x2 < 200, then the box defined by x1 < 1 and x8 >5 will be included in the contour for 2-d boxes. Often, each point on a dimension contour will be defined by different restrictions on the same set of dimensions, but this is not always true.

Dominating points are those coverage and density combinations which satisfy two conditions: They are associated with boxes generated by dropping dimensions from those boxes on the trajectory, and they lie "above and to the right" of the current trajectory. That is, they are more simply defined, and also extend the density coverage frontier. While a dominating point may not represent an overall ideal tradeoff between density, coverage and interpretability, it should be locally preferred to points nearby.

After selecting which display option they would like, the user is then asked to select candidate points from the trajectory displayed. Clicking on points displays a number for the box associated with that point. When clicking on points on the original trajectory (denoted by large filled circles), the number shown refers to the index number of exactly that box. When clicking on dominating points or points on contour lines, the number shown references the box on the trajectory from which the box represented by the point of interest was derived. That is, points not on the trajectory are all derived by taking some box on the trajectory and removing some dimension restrictions - and the number shown when you click on those points refers to the original full box. This should be born in mind later on.

After identifying candidate boxes, their statistics are displayed in the R console. The user then picks a box to consider in more detail. The program then shows various additional information about the box, and afterwards the user is given the opportunity to drop specific dimension restrictions. This completes the process of selecting a single box. The user is then given opportunity to continue

covering, in which they repeat the process above on all the data not encompassed by the previously selected box.

Value

A *box sequence* object, which is a list, each entry containing a *box* object (which, in an ideal more formalized world, would be a class). A box object contains a great deal of information about the particular box in question, including it's definition and associated statistics (described below). Additionally, the box sequence object contains two attributes, `estats` and `olap` which are ensemble statistics for the entire box sequence. While the structure of the output below may be of interest for advanced users, there is no need for the non-R user to be familiar with these outputs, as there are multiple functions for interpreting and displaying the output in a more friendly manner, such as `seqinfo` and `dimplot`.

`y.mean`	Numeric, the mean of the points inside the box. If the output data is 0-1, then this is the *density*. Note that if this is something other than the first box in the sequence, the density is contingent on the covering process up to that point.
`box`	A 2 by d matrix giving the absolute bounds of the box, including those bounds that were not restricted. For unrestricted dimension ends, they are taken from the range of the data along that dimension.
`mass`	The number of points inside the box, expressed as a fraction of the (sub)dataset used to generate this box. The "full" mass can be found in the `olap` attribute of the box sequence.
`dimlist`	A list of three logical vectors (`either`, `lower`, and `upper`), each having length equal to the number of input dimension. `upper` and `lower` indicate whether the upper and lower end were restricted, and `either` is just an OR of `lower` and `upper`. Thus, thus one way to only see the restricted box dimensions is with the code `bs[[boxnumber]]$box[,bs[[boxnumber]]$dimlist$either]`., where 'bs' is replaced by the name of the box sequence object.
`morestats`	A matrix with one row per restricted box definition, which contains the "remove variable" statistics. The columns are as follows: Column number in input matrix, density, coverage, support.
`relcoverage`	Tracks the coverage of the entire trajectory from which this box was taken, with it normalized to the subdataset the box was taken from.
`pvvalist`	A matrix giving the quasi-p-values for each dimension restriction.
`freqmat`	A matrix giving the reproducibility statistics (assuming the `sdprim` was called with argument `repro=TRUE`). The columns correspond to the columns of the input matrix, and the first row gives the reproducibility statistics when PRIM was matched on coverage, the second when it was matched on density. The entries represent the fraction of time each dimension was restricted when PRIM was rerun on `nbump` random subsamples (of size `N*dfrac`) of the dataset.
`index`	For the sake of reproducibility, this gives the index of the box in the trajectory from which it was selected. Note that, unless this is the first box in the sequence, the accuracy of this index is contingent on selection of the identical index, dimensions and restrictions for the previous boxes, and parameters for PRIM - ie, it refers to the trajectory that results after the previous boxes in the sequence have been selected.

`relbox` A matrix of structure similar to `box`, except that the bounds are normalized so that they range from zero to one. These are used in the `dimplot` command for visualizing dimension restrictions.

Box Sequence Object Attributes

The overall box sequence object returns three attributes as well. The list `estats` contains "ensemble" statistics for the entire dataset, as follows:

ecov Total coverage for the box sequence.

esup Total support for the box sequence.

eden Overall density for the space encompassed by the box sequence.

npts Total points in the dataset used.

ninter Total number of interesting points in the dataset (after thresholding).

intin Total number of interesting points captured by the box sequence.

totin Total number of points captured by the box sequence.

totdims Total dimensions in the input data.

The attribute `olaps` contains a matrix that holds several pieces of information. The diagonals display the absolute coverage of box i (in position matrix[i,i]). The lower triangle displays the the overlap in total interesting points between boxes i and j. The upper triangle displays the total points in common. All are expressed as a fraction of total points in the dataset.

Lastly, the attribute `data` contains the data used to generate the boxes, augmented by columns full of 0's and 1's. The first augmented column is named 'in.seq' and gives a 1 for any point that falls in any box in the box set, and the remaining augmented columns are titled 'in.BX', where 'X' is the box number in the box sequence. To access this data frame, use `mynewdata <- attr(myboxes, "data")`.

Author(s)

Benjamin P. Bryant, <bryant@prgs.edu>

See Also

`sd.start` for reading in and cleaning data, `seqinfo` for viewing the output of `sdprim`, and `dimplot` for visualizing dimension restrictions.

Examples

```
#Load some example data to play with:
data(quakes)
#quakes is a 1000 by 5 dataset of earthquake information.  This has no obvious
#policy significance, but we can use this built-in dataset to illustrate the use
#of PRIM.

#Here are the columns:
colnames(quakes)
```

```
#We will say magnitude is the output of interest, and call earthquakes greater
#5.0 'interesting.'  We can then call sdprim two different ways.

#First, make an input matrix from columns 1,2,3 and 5
inputs <- quakes[,c(1:3,5)]  #could also do quakes[,-4]

#Now put our unthresholded y vector:
yout <- quakes[,"mag"] #could also do quakes[,4]

#Now we can either call sdprim and threshold inside PRIM, like this:
## Not run: myboxes <- sdprim(x=inputs, y=yout, thresh=5.0, threshtype=">")

#Or we can first threshold yout:
ythresh <- 1*(yout>5.0)

#and then call sdprim without worrying about the thresholds:
## Not run: myboxes <- sdprim(x=inputs, y=ythresh)
```

`seqinfo` *Display Box Sequence Information*

Description

Function takes a box sequence as output by `sdprim`, and prints it nicely to the screen, and also to a text file if desired.

Usage

```
seqinfo(box.seq, outfile = NA)
```

Arguments

`box.seq`	A box sequence object as output by `sdprim`.
`outfile`	A character specifying a filename for outputting a copy of the printed display. The default `NA` argument will not output a file.

Value

Nothing of value returned, but outputs text to screen, and also to the text file specified in `outfile`.

Author(s)

Benjamin P. Bryant, <bryant@prgs.edu>

See Also

`sdprim`, `dimplot`

Examples

```
#Load an example box sequence for demonstration:
data(exboxes)

#Display information about the box sequence:
seqinfo(exboxes)

#Display info, and also print it out:
seqinfo(exboxes, outfile="demofile.txt")
```

Index