

Package ‘rattle’

March 7, 2010

Type Package

Title A graphical user interface for data mining in R using Gnome

Version 2.5.24

Date 2010-03-06

Author Graham Williams <Graham.Williams@togaware.com>

Maintainer Graham Williams <Graham.Williams@togaware.com>

Depends R (>= 2.8.0), pmml (>= 1.2.13)

Suggests RGtk2, ada, amap, arules, biclust, cairoDevice, car, cba, colorspace, doBy, e1071, ellipse, fEcofin, fBasics, foreign, fpc, gdata, gtools, gplots, gWidgetsRGtk2, Hmisc, kernlab, latticist, Matrix, mice, network, nnet, odfWeave, party, playwith, psych, randomForest, reshape, rggobi, ROCR, RODBC, rpart, RSvgDevice, survival, timeDate, XML

Description Rattle (the R Analytic Tool To Learn Easily) provides a Gnome (RGtk2) based interface to R functionality for data mining. The aim is to provide a simple and intuitive interface that allows a user to quickly load data from a CSV file (or via ODBC), transform and explore the data, build and evaluate models, and export models as PMML (predictive modelling markup language) or as scores. All of this with knowing little about R. All R commands are logged and commented through the log tab and so available to the user as a script file or as an aide to the user to interact directly with R itself. Rattle also exports a number of utility functions and the graphical user interface, invoked as rattle(), does not need to be run to deploy these.

License GPL (>= 2)

LazyLoad yes

LazyData yes

URL <http://rattle.togaware.com/>

Repository CRAN

Date/Publication 2010-03-07 17:09:58

R topics documented:

acquireAuditData	2
asRules	3
asRules.rpart	4
audit	5
binning	6
calcInitialDigitDistr	6
calculateAUC	7
centers.hclust	8
drawTreeNode	9
drawTreesAda	10
evaluateRisk	11
genPlotTitleCmd	12
listTreesAda	13
listVersions	14
modalvalue	14
overwritePackageFunction	15
plotBenfordsLaw	16
plotNetwork	16
plotOptimalLine	17
plotRisk	19
printRandomForests	21
randomForest2Rules	22
rattle	23
rattle.print.rpart	24
rattle.print.summary.multinom	25
savePlotToFile	25
treerset.randomForest	26
weather	27
wine	29
Index	31

acquireAuditData *Generate the audit dataset.*

Description

Rattle uses an artificial dataset for demonstration purposes. This function retrieves the source data <ftp://ftp.ics.uci.edu/pub/machine-learning-databases/adult/adult.data> and then transforms the data in a variety of ways.

Usage

```
acquireAuditData(write.to.file=FALSE)
```

Arguments`write.to.file`

Whether to generate a collection of files based on the data. The files generated include: `audit.csv`, `audit.Rdata`, `audit.arf`, and `audit_missing.csv`

Details

See the function definition for details of the processing done on the data downloaded from the UCI repository.

Value

By default the function returns a data frame containing the audit dataset. If `write.to.file` is `TRUE` then the data frame is returned invisibly.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

See Also

[audit](#), [rattle](#).

asRules

List the rules corresponding to the rpart decision tree

Description

Display a list of rules for an rpart decision tree.

Usage

```
asRules(model, compact=FALSE, ...)
```

Arguments

<code>model</code>	an rpart model.
<code>compact</code>	whether to list categorical compactly.
<code>...</code>	further arguments passed to or from other methods.

Details

Traverse a decision tree to generate the equivalent set of rules, one rule for each path from the root node to a leaf node.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

Examples

```
## Not run: asRules.rpart(my.rpart)
```

asRules.rpart	<i>List the rules corresponding to the rpart decision tree</i>
---------------	--

Description

Display a list of rules for an rpart decision tree.

Usage

```
## S3 method for class 'rpart':  
asRules(model, compact=FALSE, ...)
```

Arguments

model	an rpart model.
compact	whether to list cateogricals compactly.
...	further arguments passed to or from other methods.

Details

Traverse a decision tree to generate the equivalent set of rules, one rule for each path from the root node to a leaf node.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

Examples

```
## Not run: asRules.rpart(my.rpart)
```

`audit`*Sample dataset for illustration Rattle functionality.*

Description

The audit dataset is an artificially constructed dataset that has some of the characteristics of a true financial audit dataset for modelling productive and non-productive audits of a person's financial statement. A productive audit is one which identifies errors or inaccuracies in the information provided by a client. A non-productive audit is usually an audit which found all supplied information to be in order.

The audit dataset is used to illustrate binary classification. The target variable is identified as `TARGET_Adjusted`.

The dataset is quite small, consisting of just 2000 entities. Its primary purpose is to illustrate modelling in Rattle, so a minimally sized dataset is suitable.

The dataset itself is derived from publicly available data (which has nothing to do with audits). See [acquireAuditData](#) for details.

Format

A data frame. In line with data mining terminology we refer to the rows of the data frame (or the observations) as entities. The columns are referred to as variables. The entities represent people in this case. We describe the variables here:

`ID` This is a unique identifier for each person.

`Age` The age.

`Employment` The type of employment.

`Education` The highest level of education.

`Marital` Current marital status.

`Occupation` The type of occupation.

`Income` The amount of income declared.

`Gender` The persons gender.

`Deductions` Total amount of expenses that a person claims in their financial statement.

`Hours` The average hours worked on a weekly basis.

`IGNORE_Accounts` The main country in which the person has most of their money banked. Note that the variable name is prefixed with `IGNORE`. This is recognised by Rattle as the default role for this variable.

`RISK_Adjustment` This variable records the monetary amount of any adjustment to the person's financial claims as a result of a productive audit. This variable, which should not be treated as an input variable, is thus a measure of the size of the risk associated with the person.

`TARGET_Adjusted` The target variable for modelling (generally for classification modelling). This is a numeric field of class integer, but limited to 0 and 1, indicating non-productive and productive audits, respectively. Productive audits are those that result in an adjustment being made to a client's financial statement.

binning

Perform binning

Description

Perform binning.

Usage

```
binning(x, bins=4, method=c("quantile", "kmeans"),
        labels=NULL, ordered=TRUE)
```

Arguments

x	data.
bins	number of bins.
method	mechanism.
labels	labels.
ordered	oredered.

Details

Perform binning

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

calcInitialDigitDistr*Generate a frequency count of the initial digits*

Description

In the context of Benford's Law calculate the distribution of the frequencies of the first digit of the numbers supplied as the argument.

Usage

```
calcInitialDigitDistr(l, digit=1,
                     split=c("none", "positive", "negative"))
```

Arguments

l	a vector of numbers.
digit	the digit to generate frequencies for.
split	whether and how to split the digits.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

calculateAUC *Determine area under a curve (e.g. a risk or recall curve) of a risk chart*

Description

Given the evaluation returned by evaluateRisk, for example, calculate the area under the risk or recall curves, to use as a metric to compare the performance of a model.

Usage

```
calculateAUC(x, y)
```

Arguments

x	a vector of values for the x points.
y	a vector of values for the y points.

Details

The area is returned.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

See Also

[evaluateRisk](#).

Examples

```
## this is usually used in the context of the evaluateRisk function
## Not run: ev <- evaluateRisk(predicted, actual, risk)

## imitate this output here
ev <- data.frame(Caseload=c(1.0, 0.8, 0.6, 0.4, 0.2, 0),
                 Precision=c(0.15, 0.18, 0.21, 0.25, 0.28, 0.30),
                 Recall=c(1.0, 0.95, 0.80, 0.75, 0.5, 0.0),
                 Risk=c(1.0, 0.98, 0.90, 0.77, 0.30, 0.0))

## Calculate the areas under the Risk and the Recall curves.
calculateAUC(ev$Caseload, ev$Risk)
calculateAUC(ev$Caseload, ev$Recall)
```

centers.hclust *List Cluster Centers for a Hierarchical Cluster*

Description

Generate a matrix of centers from a hierarchical cluster.

Usage

```
centers.hclust(x, h, nclust=10, use.median=FALSE)
```

Arguments

x	The data used to build the cluster.
h	A hclust object.
nclust	Number of clusters.
use.median	Use meadion instead of mean.

Details

For the specified number of clusters, cut the hierarchical cluster appropriately to that number of clusters, and return the mean (or median) of each resulting cluster.

Author(s)

Daniele Medri and <Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

drawTreeNodes *Draw nodes of a decision tree*

Description

Draw the nodes of a decision tree

Usage

```
drawTreeNodes(tree, cex = par("cex"), pch = par("pch"),
               size = 4 * cex, col = NULL, nodeinfo = FALSE,
               units = "", cases = "obs",
               digits = getOption("digits"),
               decimals = 2,
               print.levels = TRUE, new = TRUE)
```

Arguments

tree	an rpart decision tree.
cex	.
pch	.
size	.
col	.
nodeinfo	.
units	.
cases	.
digits	.
decimals	the number of decimal digits to include in numeric split nodes.
print.levels	.
new	.

Details

A variation of draw.tree from the maptree package.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

Examples

```
## this is usually used in the context of the plotRisk function
## Not run: drawTreeNodes(ds.rp)
```

drawTreesAda	<i>Draw trees from an Ada model</i>
--------------	-------------------------------------

Description

Using the Rattle drawTreeNodes, draw a selection of Ada trees.

Usage

```
drawTreesAda(model, trees=0, title="")
```

Arguments

model	an ada model.
trees	The list of trees to draw. Use 0 to draw all trees.
title	An option title to add.

Details

Using Rattle's drawTreeNodes underneath, a plot for each of the specified trees from an Ada model will be displayed.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

Examples

```
## Not run: drawTreesAda(ds.ada)
```

`evaluateRisk`*Summarise the performance of a data mining model*

Description

By taking predicted values, actual values, and measures of the risk associated with each case, generate a summary that groups the distinct predicted values, calculating the accumulative percentage Caseload, Recall, Risk, Precision, and Measure.

Usage

```
evaluateRisk(predicted, actual, risks)
```

Arguments

<code>predicted</code>	a numeric vector of probabilities (between 0 and 1) representing the probability of each entity being a 1.
<code>actual</code>	a numeric vector of classes (0 or 1).
<code>risks</code>	a numeric vector of risk (e.g., dollar amounts) associated with each entity that has a actual of 1.

Author(s)

```
<Graham.Williams@togaware.com>
```

References

Package home page: <http://rattle.togaware.com>

See Also

[plotRisk.](#)

Examples

```
## simulate the data that is typical in data mining

## we often have only a small number of positive known case
cases <- 1000
actual <- as.integer(rnorm(cases) > 1)
adjusted <- sum(actual)
nfa <- cases - adjusted

## risks might be dollar values associated adjusted cases
risks <- rep(0, cases)
risks[actual==1] <- round(abs(rnorm(adjusted, 10000, 5000)), 2)
```

```
## our models will generated a probability of a case being a 1
predicted <- rep(0.1, cases)
predicted[actual==1] <- predicted[actual==1] + rnorm(adjusted, 0.3, 0.1)
predicted[actual==0] <- predicted[actual==0] + rnorm(nfa, 0.1, 0.08)
predicted <- signif(predicted)

## call upon evaluateRisk to generate performance summary
ev <- evaluateRisk(predicted, actual, risks)

## have a look at the first few and last few
head(ev)
tail(ev)

## the performance is usually presented as a Risk Chart
## under the CRAN MS/Windows this causes a problem, so don't run for now
## Not run: plotRisk(ev$Caseload, ev$Precision, ev$Recall, ev$Risk)
```

genPlotTitleCmd *Generate a string to add a title to a plot*

Description

Generate a string that is intended to be `eval`'d that will add a title and sub-title to a plot. The string is a call to `title`, supplying the given arguments, `paste`d together, as the main title, and generating a sub-title that begins with 'Rattle' and continues with the current date and time, and finishes with the current user's username. This is used internally in Rattle to adorn a plot with relevant information, but may be useful outside of Rattle.

Usage

```
genPlotTitleCmd(..., vector=FALSE)
```

Arguments

... one or more strings that will be `paste`d together to form the main title.
vector whether to return a vector as the result.

Author(s)

```
<Graham.Williams@togaware.com>
```

References

Package home page: <http://rattle.togaware.com>

See Also

`eval`, `title`, `plotRisk`.

Examples

```
# generate some random plot
plot(rnorm(100))

# generate the string representing the command to add titles
tl <- genPlotTitleCmd("Sample Plot of", "No Particular Importance")

# cause the string to be executed as an R command
eval(parse(text=tl))
```

listTreesAda	<i>List trees from an Ada model</i>
--------------	-------------------------------------

Description

Display the textual representation of a selection of Ada trees.

Usage

```
listTreesAda(model, trees=0)
```

Arguments

model	an ada model.
trees	The list of trees to list. Use 0 to list all trees.

Details

Using rpart's print method display each of the specified trees from an Ada model.

Author(s)

```
<Graham.Williams@togaware.com>
```

References

Package home page: <http://rattle.togaware.com>

Examples

```
## Not run: listTreesAda(ds.ada)
```

listVersions	<i>Versions of Installed Packages</i>
--------------	---------------------------------------

Description

Generate a list of packages installed and their version number.

Usage

```
listVersions(file="", ...)
```

Arguments

file	a character string naming a file or a connection open for writing. "" indicates output to the console.
...	arguments to write.csv .

Details

This function is useful in reporting problems or bugs, to ensure there is a clear match of R package versions between the system exhibiting the issue and the test system replicating the issue.

By default the information is written to the console in a comma separated form, that is ideally designed to be written to a CSV file for emailing.

Author(s)

<Graham.Williams@togaware.com>

See Also

[write.csv](#)

modalvalue	<i>Calculate the mode of a vector, array or list.</i>
------------	---

Description

The mode is the most common or modal value of a list. This function calculates the mode of a vector, array or list (lists are flattened).

Usage

```
modalvalue(x, na.rm=FALSE)
```

Arguments

<code>x</code>	A vector, array or list.
<code>na.rm</code>	Whether to remove missing values.

Details

Rattle needed this function, and R does not provide one. This version came from the R Wiki.

References

R Wiki: <http://wiki.r-project.org/rwiki/doku.php?id=tips:stats-basic:modalvalue>

`overwritePackageFunction`

Overwrite an internal function of a package.

Description

Rattle supports a plugin infrastructure to allow other package writers to fine tune the Rattle interface and behaviour. This function provides a simple mechanism to allow these packages to define their own versions of functions that will replace the same function in Rattle.

Usage

```
overwritePackageFunction(fname, fun, pkg)
```

Arguments

<code>fname</code>	a string naming the Rattle function to overwrite.
<code>fun</code>	the actual function to overwrite the Rattle function.
<code>pkg</code>	the package in which the function will be overwritten.

Details

Refer to the Rattle home page in the URL below for a growing reference manual for using Rattle, which `overwritePackageFunction` is part of.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

See Also

[rattle.](#)

plotBenfordsLaw *Plot a chart comparing Benford's Law with a supplied numeric vector*

Description

Plots a barchart of Benford's Law and the distribution of the frequencies of the first digit of the numbers supplied as the argument.

Usage

```
plotBenfordsLaw(l)
```

Arguments

l a vector of numbers to compare to Benford's Law.

Author(s)

```
<Graham.Williams@togaware.com>
```

References

Package home page: <http://rattle.togaware.com>

Examples

```
# A simple example using the audit data from Rattle.
data(audit)
plotBenfordsLaw(audit$Income)
```

plotNetwork *Plot a circular map of network links between entities*

Description

Plots a circular map of entities and their relationships. The entities are around the edge of the circle with lines linking the entities depending on their relationships as represented in the supplied FLOW argument. Line widths represent relative magnitude of flows, as do line colours, and the font size of a label for an entity represents the size of the total flow into that entity. Useful for displaying, for example, cash flows between entities.

Usage

```
plotNetwork(flow)
```

Arguments

flow a square matrix of directional flows.

Details

The FLOW is a square matrix that records the directional flow and magnitude of the flow from one entity to another. The flow may represent a flow of cash from one company to another company. The dimensions of the square matrix are the number of entities represented.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

Examples

```
# Create a small sample matrix.
flow <- matrix(c(0, 10000, 0, 0, 20000,
                1000, 0, 10000, 0, 0,
                5000, 0, 0, 0, 3000,
                0, 1000000, 600000, 0, 0,
                0, 50000, 0, 500000, 0),
              nrow=5, byrow=TRUE)
rownames(flow) <- colnames(flow) <- c("A", "B", "C", "D", "E")
plotNetwork(flow)

# Use data from the network package.
data(flo)
plotNetwork(flo)
```

plotOptimalLine *Plot three lines on a risk chart, one vertical and two horizontal*

Description

Plots a vertical line at x up to max of y1 and y2, then horizontal from this line at y1 and y2. Intended for plotting on a plotRisk.

Usage

```
plotOptimalLine(x, y1, y2, pr = NULL, colour = "plum", label = NULL)
```

Arguments

x	location of vertical line.
y1	location of one horizontal line.
y2	location of other horizontal line.
pr	A print a percentage at this point.
colour	of the line.
label	at bottom of line.

Details

Intended to plot an optimal line on a Risk Chart as plotted by `plotRisk`.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

See Also

[plotRisk](#).

Examples

```
## this is usually used in the context of the plotRisk function
## Not run: ev <- evaluateRisk(predicted, actual, risk)

## imitate this output here
ev <- NULL
ev$Caseload <- c(1.0, 0.8, 0.6, 0.4, 0.2, 0)
ev$Precision <- c(0.15, 0.18, 0.21, 0.25, 0.28, 0.30)
ev$Recall <- c(1.0, 0.95, 0.80, 0.75, 0.5, 0.0)
ev$Risk <- c(1.0, 0.98, 0.90, 0.77, 0.30, 0.0)

## plot the Risk Chart
plotRisk(ev$Caseload, ev$Precision, ev$Recall, ev$Risk,
         chosen=60, chosen.label="Pr=0.45")

## plot the optimal point
plotOptimalLine(40, 77, 75, colour="maroon")
```

plotRisk

*Plot a risk chart***Description**

Plots a Rattle Risk Chart. Such a chart has been developed in a practical context to present the performance of data mining models to clients, plotting a caseload against performance, allowing a client to see the tradeoff between coverage and performance.

Usage

```
plotRisk(cl, pr, re, ri = NULL, title = NULL,
         show.legend = TRUE, xleg = 20, yleg = 16,
         optimal = NULL, optimal.label = "", chosen = NULL, chosen.label = "",
         include.baseline = TRUE, dev = "", filename = "", show.knots = NULL,
         show.lift=TRUE, show.precision=TRUE,
         risk.name = "Revenue", recall.name = "Adjustments",
         precision.name = "Strike Rate")
```

Arguments

cl	a vector of caseloads corresponding to different probability cutoffs. Can be either percentages (between 0 and 100) or fractions (between 0 and 1).
pr	a vector of precision values for each probability cutoff. Can be either percentages (between 0 and 100) or fractions (between 0 and 1).
re	a vector of recall values for each probability cutoff. Can be either percentages (between 0 and 100) or fractions (between 0 and 1).
ri	a vector of risk values for each probability cutoff. Can be either percentages (between 0 and 100) or fractions (between 0 and 1).
title	the main title to place at the top of the plot.
show.legend	whether to display the legend in the plot.
xleg	the x coordinate for the placement of the legend.
yleg	the y coordinate for the placement of the legend.
optimal	a caseload (percentage or fraction) that represents an optimal performance point which is also plotted. If instead the value is TRUE then the optimal point is identified internally (maximum value for $(\text{recall-caseload}) + (\text{risk-caseload})$) and plotted.
optimal.label	a string which is added to label the line drawn as the optimal point.
chosen	a caseload (percentage or fraction) that represents a user chosen optimal performance point which is also plotted.
chosen.label	a string which is added to label the line drawn as the chosen point.
include.baseline	if TRUE (the default) then display the diagonal baseline.

<code>dev</code>	a string which, if supplied, identifies a device type as the target for the plot. This might be one of <code>wmf</code> (for generating a Windows Metafile, but only available on MS/Windows), <code>pdf</code> , or <code>png</code> .
<code>filename</code>	a string naming a file. If <code>dev</code> is not given then the filename extension is used to identify the image format as one of those recognised by the <code>dev</code> argument.
<code>show.knots</code>	a vector of caseload values at which a vertical line should be drawn. These might correspond, for example, to individual paths through a decision tree, illustrating the impact of each path on the caseload and performance.
<code>show.lift</code>	whether to label the right axis with lift.
<code>show.precision</code>	whether to show the precision plot.
<code>risk.name</code>	a string used within the plot's legend that gives a name to the risk. Often the risk is a dollar amount at risk from a fraud or from a bank loan point of view, so the default is <code>Revenue</code> .
<code>recall.name</code>	a string used within the plot's legend that gives a name to the recall. The recall is often the percentage of cases that are positive hits, and in practise these might correspond to known cases of fraud or reviews where some adjustment to perhaps a incom tax return or application for credit had to be made on reviewing the case, and so the default is <code>Adjustments</code> .
<code>precision.name</code>	a string used within the plot's legend that gives a name to the precision. A common name for precision is <code>Strike Rate</code> , which is the default here.

Details

Caseload is the percentage of the entities in the dataset covered by the model at a particular probability cutoff, so that with a cutoff of 0, all (100%) of the entities are covered by the model. With a cutoff of 1 (0%) no entities are covered by the model. A diagonal line is drawn to represent a baseline random performance. Then the percentage of positive cases (the recall) covered for a particular caseload is plotted, and optionally a measure of the percentage of the total risk that is also covered for a particular caseload may be plotted. Such a chart allows a user to select an appropriate tradeoff between caseload and performance. The charts are similar to ROC curves. The precision (i.e., strike rate) is also plotted.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

See Also

[evaluateRisk](#), [genPlotTitleCmd](#).

Examples

```
## this is usually used in the context of the evaluateRisk function
## Not run: ev <- evaluateRisk(predicted, actual, risk)

## imitate this output here
ev <- NULL
ev$Caseload <- c(1.0, 0.8, 0.6, 0.4, 0.2, 0)
ev$Precision <- c(0.15, 0.18, 0.21, 0.25, 0.28, 0.30)
ev$Recall <- c(1.0, 0.95, 0.80, 0.75, 0.5, 0.0)
ev$Risk <- c(1.0, 0.98, 0.90, 0.77, 0.30, 0.0)

## plot the Risk Chart
plotRisk(ev$Caseload, ev$Precision, ev$Recall, ev$Risk,
         chosen=60, chosen.label="Pr=0.45")

## Add a title
eval(parse(text=genPlotTitleCmd("Sample Risk Chart")))
```

```
printRandomForests Print a representtaion of the Random Forest models to the console
```

Description

A randomForest model, by default, consists of 500 decision trees. This function walks through each tree and generates a set of rules which are printed to the console. This takes a considerable amount of time and is provided for users to access the actual model, but it is not yet used within the Rattle GUI. It may be used to display the output of the RF (but it takes longer to generate than the model itself!). Or it might only be used on export to PMML or SQL.

Usage

```
printRandomForests(model, models=NULL, include.class=NULL, format="")
```

Arguments

model	a randomForest model.
models	a list of integers limiting the models in MODEL that are displayed.
include.class	limit the output to the specific class.
format	possible values are "VB".

Author(s)

```
<Graham.Williams@togaware.com>
```

References

Package home page: <http://rattle.togaware.com>

Examples

```
## Display a ruleset for a specific model amongst the 500.
## Not run: printRandomForests(rfmodel, 5)

## Display a ruleset for specific models amongst the 500.
## Not run: printRandomForests(rfmodel, c(5,10,15))

## Display a ruleset for each of the 500 models.
## Not run: printRandomForests(rfmodel)
```

randomForest2Rules *Generate accessible data structure of a randomForest model*

Description

A randomForest model, by default, consists of 500 decision trees. This function walks through each tree and generates a set of rules. This takes a considerable amount of time and is provided for users to access the actual model, but it is not yet used within the Rattle GUI. It may be used to display the output of the RF (but it takes longer to generate than the model itself!). Or it might only be used on export to PMML or SQL.

Usage

```
randomForest2Rules(model, models=NULL)
```

Arguments

model a randomForest model.
models a list of integers limiting the models in MODEL that are converted.

Author(s)

```
<Graham.Williams@togaware.com>
```

References

Package home page: <http://rattle.togaware.com>

Examples

```
## Generate a ruleset for a specific model amongst the 500.
## Not run: randomForest2Rules(rfmodel, 5)

## Generate a ruleset for specific models amongst the 500.
## Not run: randomForest2Rules(rfmodel, c(5,10,15))

## Generate a ruleset for each of the 500 models.
## Not run: randomForest2Rules(rfmodel)
```

`rattle`*Display the Rattle User Interface*

Description

The Rattle user interface uses the RGtk2 package to present an intuitive point and click interface for data mining, extensively building on the excellent collection of R packages for data manipulation, exploration, analysis, and evaluation.

Usage

```
rattle(csvname=NULL)
```

Arguments

`csvname` the name of a CSV file to load into Rattle on startup.

Details

Refer to the Rattle home page in the URL below for a growing reference manual for using Rattle.

Whilst the underlying functionality of Rattle is built upon a vast collection of other R packages, Rattle itself provides a collection of utility functions used within Rattle. These are made available through loading the rattle package into your R library. The See Also section lists these utility functions that may be useful outside of Rattle.

Rattle can initialise some options using a .Rattle file if the folder in which Rattle is started. The currently supported options are .RATTLE.DATA, .RATTLE.SCORE.IN, and .RATTLE.SCORE.OUT.

If the environment variable RATTLE_DATA is defined then that is set as the default CSV file name to load. Otherwise, if .RATTLE.DATA is defined then that will be used as the CSV file to load. Otherwise, if csvname is provided then that will be used.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

See Also

[evaluateRisk](#), [genPlotTitleCmd](#), [plotRisk](#).

Examples

```
# You can start rattle with a path to a csv file to pre-specify the
# dataset. You then need to click Execute to load the data.
```

```
rattle(system.file("csv", "weather.csv", package = "rattle"))
```

rattle.print.rpart *Modified printing of rpart decision tree*

Description

Display a textual view of an rpart decision tree.

Usage

```
rattle.print.rpart(x, minlength = 0, spaces = 2, cp,
                  digits = getOption("digits"), ...)
```

Arguments

x	An rpart object.
minlength	The minimum length.
spaces	How much to indent.
cp	Complexity parameter
digits	Number of digits to display for numbers.
...	Other arguments.

Details

Print decision tree.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

```
rattle.print.summary.multinom
```

Print information about a multinomial model

Description

Displays a textual review of the performance of a multinom model.

Usage

```
rattle.print.summary.multinom(x, digits = x$digits, ...)
```

Arguments

x	An rpart object.
digits	Number of digits to print for numbers.
...	Other arguments.

Details

Print a summary of a multinom model. This is simply a modification of the print.summary.multinom function to add the number of entities!

Author(s)

```
<Graham.Williams@togaware.com>
```

References

Package home page: <http://rattle.togaware.com>

```
savePlotToFile
```

Save a plot in some way

Description

For the current device, or for the device identified, save the plot displayed there in some way. This is either saved to file, copied to the clipboard for pasting into other applications, or sent to the printer for saving a hard copy.

Usage

```
savePlotToFile(file.name, dev.num=dev.cur())  
copyPlotToClipboard(dev.num=dev.cur())  
printPlot(dev.num=dev.cur())
```

Arguments

file.name	Character string naming the file including the file name extension which is used to specify the type of file to save.
dev.num	A device number indicating which device to save.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

treeset.randomForest

Generate a representation of a tree in a Random Forest

Description

Often we want to view the actual trees built by a random forest. Although reviewing all 500 trees might be a bit much, this function allows us to at least list them.

Usage

```
treeset.randomForest(model, n=1, root=1, format="R")
```

Arguments

model	a randomForest model.
n	a specific tree to list.
root	where to start the stree from, primarily for internal use.
format	one of "R", "VB".

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: <http://rattle.togaware.com>

Examples

```
## Display a treeset for a specific model amongst the 500.  
## Not run: treeset.randomForests(rfmodel, 5)
```

 weather

Sample and full dataset for illustrating Rattle functionality.

Description

Weather observations from a number of locations around Australia, obtained from the Australian Commonwealth Bureau of Meteorology and processed to create a sample dataset for illustrating data mining using R and Rattle.

The data has been processed to provide a target variable `RainTomorrow` (whether there is rain on the following day - No/Yes) and a risk variable `RISK_MM` (how much rain). Various transformations are performed on the data.

The `weather` dataset is a fixed subset of the full `weatherAUS` dataset, for the purpose of repeatable demonstration. It includes only one year of data for a single weather station.

The `weatherAUS` dataset is ever changing, with each release of Rattle. It is updated from the Bureau of Meteorology web site regularly.

The source dataset is Copyright by the Australian Commonwealth Bureau of Meteorology and used with permission.

Usage

```
weather
weatherAUS
```

Format

The `weather` dataset is a data frame containing one year of daily observations from a single weather station (Canberra). The `weatherAUS` dataset is a data frame containing over 10,000 daily observations from numerous weather stations.

`Date` Date of observation (a Date object).

`Location` Common name of the location of the weather station.

`MinTemp` Minimum temperature in degrees celcius.

`MaxTemp` Maximum temperature in degrees celcius.

`Rainfall` Rainfall for the day in mm.

`Evaporation` "Class A" pan evaporation (mm) in the 24 hours to 9am.

`Sunshine` Hours of bright sunshine in the day.

`MaxWindSpeed` Speed (km/hr) of strongest wind gust in the 24 hours to midnight.

`Temp9am` Temperature (degrees C) at 9am.

`RelHumid9am` Relative humidity (percent) at 9am.

`Cloud9am` Fraction of sky obscured by cloud at 9am. This is measured in "oktas", which are a unit of eighths. It records how many eighths of the sky are obscured by cloud. A 0 measure indicates completely clear sky whilst an 8 indicates that it is completely overcast.

WindSpeed9am Wind speed (km/hr) averaged over 10 minutes prior to 9am.
 Pressure9am Atmospheric pressure (hpa) reduced to mean sea level at 9am.
 Temp3pm Temperature (degrees C) at 3pm.
 RelHumid3pm Relative humidity (percent) at 3pm.
 Cloud3pm Fraction of sky obscured by cloud (in "oktas": eighths) at 3pm. See Cload9am for a description of the values.
 WindSpeed3pm Wind speed (km/hr) averaged over 10 minutes prior to 3pm.
 Pressure3pm Atmospheric pressure (hpa) reduced to mean sea level at 3pm.
 ChangeTemp Change in temperature.
 ChangeTempDir Direction of change in temperature.
 ChangeTempMag Magnitude of change in temperature.
 ChangeWindDirect Direction of wind change.
 MaxWindPeriod Period of maximum wind.
 RainToday Integer: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0.
 TempRange Difference between minimum and maximum temperatures (degrees C) in the 24 hours to 9am.
 PressureChange Change in pressure.
 RISK_MM The amount of rain. A kind of measure of the "risk".
 RainTomorrow The target variable. Did it rain tomorrow?

Author(s)

<Graham.Williams@togaware.com>

Source

Observations were drawn from numerous weather stations. The daily observations are available from <http://www.bom.gov.au/climate/data>. Copyright Commonwealth of Australia 2010, Bureau of Meteorology.

Definitions adapted from <http://www.bom.gov.au/climate/dwo/IDCJDW0000.shtml>

References

Package home page: <http://rattle.togaware.com>. Data source: <http://www.bom.gov.au/climate/dwo/> and <http://www.bom.gov.au/climate/data>.

See Also

[audit](#).

wine

The wine dataset from the UCI Machine Learning Repository.

Description

The wine dataset contains the results of a chemical analysis of wines grown in a specific area of Italy. Three types of wine are represented in the 178 samples, with the results of 13 chemical analyses recorded for each sample. The `Type` variable has been transformed into a categorical variable.

The data contains no missing values and consists of only numeric data, with a three class target variable (`Type`) for classification.

Usage

wine

Format

A data frame containing 178 observations of 13 variables.

`Type` The type of wine, into one of three classes, 1 (59 obs), 2(71 obs), and 3 (48 obs).

`Alcohol` Alcohol

`Malic` Malic acid

`Ash` Ash

`Alcalinity` Alcalinity of ash

`Magnesium` Magnesium

`Phenols` Total phenols

`Flavanoids` Flavanoids

`Nonflavanoids` Nonflavanoid phenols

`Proanthocyanins` Proanthocyanins

`Color` Color intensity.

`Hue` Hue

`Dilution` D280/OD315 of diluted wines.

`Proline` Proline

Source

The data was downloaded from the UCI Machine Learning Repository.

It was read as a CSV file with no header using `read.csv`. The columns were then given the appropriate names using `colnames` and the `Type` was transformed into a factor using `as.factor`. The compressed R data file was saved using `save`:

```
UCI <- "http://archive.ics.uci.edu/ml"
REPOS <- "machine-learning-databases"
wine.url <- sprintf("
wine <- read.csv(wine.url, header=FALSE)
colnames(wine) <- c('Type', 'Alcohol', 'Malic', 'Ash',
                   'Alcalinity', 'Magnesium', 'Phenols',
                   'Flavanoids', 'Nonflavanoids',
                   'Proanthocyanins', 'Color', 'Hue',
                   'Dilution', 'Proline')
wine$Type <- as.factor(wine$Type)
save(wine, file="wine.Rdata", compress=TRUE)
```

References

Asuncion, A. & Newman, D.J. (2007). *UCI Machine Learning Repository* [<http://www.ics.uci.edu/~mllearn/MLRepository.html>]. Irvine, CA: University of California, School of Information and Computer Science.

Index

- *Topic **aplot**
 - genPlotTitleCmd, 11
- *Topic **cluster**
 - centers.hclust, 8
- *Topic **datasets**
 - audit, 4
 - weather, 26
 - wine, 28
- *Topic **dplot**
 - evaluateRisk, 10
- *Topic **environment**
 - overwritePackageFunction, 14
 - rattle, 22
- *Topic **hplot**
 - calcInitialDigitDistr, 6
 - calculateAUC, 7
 - drawTreeNodes, 8
 - drawTreesAda, 9
 - listTreesAda, 12
 - modalvalue, 14
 - plotBenfordsLaw, 15
 - plotNetwork, 16
 - plotOptimalLine, 17
 - plotRisk, 18
 - printRandomForests, 20
 - randomForest2Rules, 21
 - savePlotToFile, 24
 - treerset.randomForest, 25
- *Topic **tree**
 - asRules, 3
 - asRules.rpart, 3
 - rattle.print.rpart, 23
- acquireAuditData, 2, 4
- as.factor, 28
- asRules, 3
- asRules.rpart, 3
- audit, 2, 4, 27
- binning, 5
- calcInitialDigitDistr, 6
- calculateAUC, 7
- centers.hclust, 8
- colnames, 28
- copyPlotToClipboard
 - (savePlotToFile), 24
- drawTreeNodes, 8
- drawTreesAda, 9
- eval, 11, 12
- evaluateRisk, 7, 10, 20, 23
- genPlotTitleCmd, 11, 20, 23
- listTreesAda, 12
- listVersions, 13
- modalvalue, 14
- overwritePackageFunction, 14
- paste, 11
- plotBenfordsLaw, 15
- plotNetwork, 16
- plotOptimalLine, 17
- plotRisk, 11, 12, 17, 18, 23
- printPlot (savePlotToFile), 24
- printRandomForests, 20
- randomForest2Rules, 21
- rattle, 2, 15, 22
- rattle.print.rpart, 23
- rattle.print.summary.multinom, 24
- read.csv, 28
- save, 28
- savePlotToFile, 24
- title, 11, 12
- treerset.randomForest, 25

weather, [26](#)
weatherAUS (*weather*), [26](#)
wine, [28](#)
write.csv, [13](#)