Package 'rattle'

December 16, 2019

Type Package

Title Graphical User Interface for Data Science in R

Version 5.3.0

Date 2019-12-16

Depends R (>= 3.5.0)

Imports stats, utils, ggplot2, grDevices, graphics, magrittr, methods, stringi, stringr, tidyr, dplyr, XML, rpart.plot

Suggests pmml (>= 1.2.13), bitops, colorspace, ada, amap, arules, arulesViz, biclust, cairoDevice, cba, cluster, corrplot, descr, doBy, e1071, ellipse, fBasics, foreign, fpc, gdata, ggdendro, ggraptR, gplots, grid, gridExtra, gtools, gWidgetsRGtk2, hmeasure, Hmisc, kernlab, Matrix, mice, nnet, party, plyr, psych, randomForest, RColorBrewer, readxl, reshape, rggobi, RGtk2, ROCR, RODBC, rpart, scales, SnowballC, survival, timeDate, tm, verification, wskm, xgboost

Description The R Analytic Tool To Learn Easily (Rattle) provides a collection of utilities functions for the data scientist. A Gnome (RGtk2) based graphical interface is included with the aim to provide a simple and intuitive introduction to R for data science, allowing 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. A key aspect of the GUI is that all R commands are logged and commented through the log tab. This can be saved as a standalone R script file and as an aid for the user to learn R or to copy-and-paste directly into R itself.

License GPL (>= 2)

LazyLoad yes

LazyData yes

URL https://rattle.togaware.com/

NeedsCompilation no

Author Graham Williams [aut, cph, cre],
Mark Vere Culp [cph],
Ed Cox [ctb],
Anthony Nolan [ctb],
Denis White [cph],
Daniele Medri [ctb],
Akbar Waljee [ctb] (OOB AUC for Random Forest),
Brian Ripley [cph] (print.summary.nnet),
Jose Magana [ctb] (ggpairs plots),
Surendra Tipparaju [ctb] (initial RevoScaleR/XDF),
Durga Prasad Chappidi [ctb] (initial RevoScaleR/XDF),
Dinesh Manyam Venkata [ctb] (initial RevoScaleR/XDF),
Mrinal Chakraborty [ctb] (initial RevoScaleR/XDF),
Fang Zhou [ctb] (initial xgboost),
Cameron Chisholm [ctb] (risk plot on risk chart)

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

Repository CRAN

Date/Publication 2019-12-16 06:20:03 UTC

R topics documented:

`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 colleciton 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: `https://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

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: https://rattle.togaware.com

Examples

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

asRules.rpart *List the rules corresponding to the rpart decision tree*

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: https://rattle.togaware.com

Examples

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

audit *Sample dataset to illustrate 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).

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 refered 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 over numeric data*

Description

Perform binning.

Usage

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

Arguments

`x`	the numeric data to bin.
`bins`	the number of bins to use.
`method`	whether to use "quantile", weighted quantile "wtd.quantile" or "kmeans" binning.
`labels`	the labels or names to use for each of the bins.
`ordered`	whether to build an ordered factor or not.
`weights`	vector of numeric weights for each observation for weighted quantile binning.

Details

Bin the provided nmeric data into the specified number of bins using one of the supported methods. The bins will have the names specified by labels, if supplied. The result can optionally be an ordered factor.

Value

A factor is returned.

Author(s)

Daniele Medri and Graham Williams

References

Package home page: https://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, len=1,
 sp=c("none", "positive", "negative"))
```

Arguments

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

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: https://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: https://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 unde 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, object, nclust=10, use.median=FALSE)
```

Arguments

x	The data used to build the cluster.
object	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: https://rattle.togaware.com

`comcat` *Echo data in a human readable form.*

Description

Format data in the most appropriate human readable form.

Usage

```
comcat(x, ...)
```

Arguments

x	object.
...	additional arguments passed on to format.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: https://rattle.togaware.com

Examples

```
comcat(dim(iris))
```

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>, Denis White

References

Package home page: https://rattle.togaware.com

Examples

```
## this is usually used in the context of the plotRisk function
## Not run: drawTreeNodes(rpart(Species ~ ., iris))
```

---

drawTreesAda *Draw trees from an Ada model*

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: https://rattle.togaware.com

Examples

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

errorMatrix *Generate an error matrix from actua and predicted data.*

Description

An error matrix reports the true/false potisitve/negative rates.

Usage

```
errorMatrix(actual,
                        predicted,
                        percentage=TRUE,
                        digits=ifelse(percentage,1,3),
                        count=FALSE)
```

Arguments

actual	a vector of true values.
predicted	a vector of predicted values.
percentage	return percentages.
digits	the number of digits to round results.
count	return counts.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: https://rattle.togaware.com

Examples

```
## Not run: errorMatrix(model)
```

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

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

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: https://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)
```

`fancyRpartPlot` *A wrapper for plotting rpart trees using prp*

Description

Plots a fancy RPart decision tree using the pretty rpart plotter.

Usage

```
fancyRpartPlot(model, main="", sub, caption, palettes, type=2, ...)
```

Arguments

`model`	an rpart object.
`main`	title for the plot.
`sub`	sub title for the plot. The default is a Rattle string with date, time and username.
`caption`	caption for bottom right of plot.
`palettes`	a list of sequential palettes names. As supported by RColorBrewer::brewer.pal the available names are Blues BuGn BuPu GnBu Greens Greys Oranges OrRd PuBu PuBuGn PuRd Purples RdPu Reds YlGn YlGnBu YlOrBr YlOrRd.
`type`	the type of plot to generate (2).
`...`	additional arguments passed on to prp.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: https://rattle.togaware.com

Examples

```
## Use rpart to build a decision tree.

## Not run: library(rpart)

## Set up the data for modelling.

set.seed(42)
ds     <- weather
target <- "RainTomorrow"
risk   <- "RISK_MM"
ignore <- c("Date", "Location", risk)
vars   <- setdiff(names(ds), ignore)
nobs   <- nrow(ds)
form   <- formula(paste(target, "~ ."))
train  <- sample(nobs, 0.7*nobs)
test   <- setdiff(seq_len(nobs), train)
actual <- ds[test, target]
risks  <- ds[test, risk]

# Fit the model.

fit <- rpart(form, data=ds[train, vars])

## Plot the model.

fancyRpartPlot(fit)

## Choose different colours.

fancyRpartPlot(fit, palettes=c("Greys", "Oranges"))

## Add a main title to the plot.

fancyRpartPlot(fit, main=target)


## End(Not run)
```

`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 pasted 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: https://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))
```

---

ggVarImp *Model.*

Description

Model.

Usage

```
ggVarImp(model, ...)
```

Arguments

model	object.
...	arguments passed on.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: https://rattle.togaware.com

Examples

```
## Not run: ggVarImp(model)
```

`listAdaVarsUsed` *List the variables used by an adaboost model*

Description

Returns a list of the variables used and their frequencies.

Usage

```
listAdaVarsUsed(model)
```

Arguments

`model`	an rpart object.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: https://rattle.togaware.com

listTreesAda *List trees from an Ada model*

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: https://rattle.togaware.com

Examples

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

listVersions *Versions of Installed Packages*

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 *Calculate the mode of a vector, array or list.*

Description

The mode is the most common or modal value of a list.

Usage

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

Arguments

x	A vector, array or list.
na.rm	Whether to remove missing values.

Details

This function calculates the mode of a vector, array or list (lists are flattened). This code originated from an anonymous post on the R Wiki.

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

Description

Plots a 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	Aprint 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: https://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 = 60, yleg = 55,
    optimal = NULL, optimal.label = "", chosen = NULL, chosen.label = "",
    include.baseline = TRUE, dev = "", filename = "", show.knots = NULL,
    show.lift=TRUE, show.precision=TRUE,
    risk.name = "Risk", recall.name = "Recall",
    precision.name = "Precision")
```

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 valud for (recall-casload)+(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.

`dev` a string which, if supplied, identifies a device type as the target for the plot. This might be one of `wmf` (for generating a Windows Metafile, but only available on MS/Windows), `pdf`, or `png`.

`filename` a string naming a file. If `dev` is not given then the filename extension is used to identify the image format as one of those recognised by the `dev` argument.

`show.knots` 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.

`show.lift` whether to label the right axis with lift.

`show.precision` whether to show the precision plot.

`risk.name` 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 `Revenue`.

`recall.name` 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 `Adjustments`.

`precision.name` a string used within the plot's legend that gives a name to the precision. A common name for precision is `Strike Rate`, 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: https://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 representation 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: https://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: https://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 by very many authors for data manipulation, exploration, analysis, and evaluation.

Usage

```
rattle(csvname=NULL, dataset=NULL, useGtkBuilder=TRUE)
```

Arguments

csvname	the optional name of a CSV file to load into Rattle on startup.
dataset	The optional name as a character string of a dataset to load into Rattle on startup.
useGtkBuilder	if not supplied then automatically determine whether to use the new GtkBuilder rather than the deprecated libglade. A user can override the heuristic choice with TRUE or FALSE.

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.

Two environments are exported by Rattle, capturing the current rattle state (crs) and the current rattle variables (crv).

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: https://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.

## Not run: rattle(system.file("csv", "weather.csv", package = "rattle"))
```

---

rattle.print.summary.multinom

Print information about a multinomial model

Description

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

Usage

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

Arguments

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

Details

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

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: https://rattle.togaware.com

rattleInfo *Extract Rattle and related package information.*

Description

Display system information, including versions of Rattle and R, operating system, and versions of other packages used by Rattle. Useful for reporting bugs but also invisibly returns a list of packages that have updates available and can be passed to install.packages().

Usage

```
rattleInfo(all.dependencies=FALSE,
           include.not.installed=FALSE,
           include.not.available=FALSE,
           include.libpath=FALSE)
```

Arguments

`all.dependencies`
: If TRUE then check the full dependency graph for Rattle and list all of those packages (which may take quite a few seconds to compute), or else just list those key packages that Rattle Depends on and Suggests.

`include.not.installed`
: If TRUE then make mention of any packages that are not installed, but are available.

`include.not.available`
: If TRUE then make mention of any packages that are not available from CRAN.

`include.libpath`
: If TRUE then list the library location where each package is installed.

Details

This is a support function to list useful information to provide the developers with information about the system environment when running Rattle. It is intended to provide the information that is useful in reporting bugs.

It also lists the currently installed version of a number of packages that Rattle makes use of as well as checking for any updates available for those packages.

If updates are found then a command is generated and printed so that a user can simply copy and paste the command to update the relevant packages. The function also invisibly returns the list of packages that can be updated, so that we can do something like: install.packages(rattleInfo()).

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: https://rattle.togaware.com

See Also

rattle.

---

rescale.by.group *Transform a numeric vector by grouping it according to the values of the supplied factor and then rescaling within the groups.*

Description

The numeric vector is remapped to integers from 0 to max-1, with any missing values mapped to the midpoint. Original idea from Tony Nolan. This will eventually be generalised to do the remapping using any of the rescaling functions.

Usage

```
rescale.by.group(x, by=NULL, type = "irank", itop = 100)
```

Arguments

x	The numeric vector to rescale.
by	A factor of the same length as x used to define the groups.
type	The type of rescaling to perform.
itop	For an integer remapping this is the number of groups, so that the numeric values are maped to the integers from 0 to (max-1).

Details

This Rattle support function, which is also useful by itself, provides a simple mechanism to rescale a numeric variable. Several rescalings are possible. The rescaling is done by first grouping the observations according to the by argument.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: https://rattle.togaware.com

See Also

rattle.

riskchart *Plot a risk chart*

Description

Plots a Rattle Risk Chart for binary classification models using ggplot2. 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

```
riskchart(pr,
          ac,
          ri               = NULL,
          title            = "Risk Chart",
          title.size       = 10,
          subtitle         = NULL,
          caption          = TRUE,
          show.legend      = TRUE,
          optimal          = NULL,
          optimal.label    = "",
          chosen           = NULL,
          chosen.label     = "",
          include.baseline = TRUE,
          dev              = "",
          filename         = "",
          show.knots       = NULL,
          show.lift        = TRUE,
          show.precision   = TRUE,
          show.maximal     = TRUE,
          risk.name        = "Risk",
          recall.name      = "Recall",
          precision.name   = "Precision",
          thresholds       = NULL,
          legend.horiz     = TRUE)
```

Arguments

pr	The predicted class for each observation.
ac	The actual class for each observation.
ri	The risk class for each observation.
title	the main title to place at the top of the plot.
title.size	font size for the main title.
subtitle	subtitle under the main title.

caption	caption for the bottom right of plot.
show.legend	whether to display the legend in the plot.
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 valud for (recall-casload)+(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.
dev	a string which, if supplied, identifies a device type as the target for the plot. This might be one of wmf (for generating a Windows Metafile, but only available on MS/Windows), pdf, or png.
filename	a string naming a file. If dev is not given then the filename extension is used to identify the image format as one of those recognised by the dev argument.
show.knots	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.
show.lift	whether to label the right axis with lift.
show.precision	whether to show the precision plot.
show.maximal	whether to show the maximal performance line.
risk.name	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 Revenue.
recall.name	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 Adjustments.
precision.name	a string used within the plot's legend that gives a name to the precision. A common name for precision is Strike Rate, which is the default here.
thresholds	whether to display scores along the top axis.
legend.horiz	whether to display a horizontal legend.

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: https://rattle.togaware.com

See Also

evaluateRisk, genPlotTitleCmd.

Examples

```
## Not run:

## Use rpart to build a decision tree.

library(rpart)

## Set up the data for modelling.

set.seed(42)
ds     <- weather
target <- "RainTomorrow"
risk   <- "RISK_MM"
ignore <- c("Date", "Location", risk)
vars   <- setdiff(names(ds), ignore)
nobs   <- nrow(ds)
form   <- formula(paste(target, "~ ."))
train  <- sample(nobs, 0.7*nobs)
test   <- setdiff(seq_len(nobs), train)
actual <- ds[test, target]
risks  <- ds[test, risk]

# Build the model.

model <- rpart(form, data=ds[train, vars])

## Obtain predictions.

predicted <- predict(model, ds[test, vars], type="prob")[,2]

## Plot the Risk Chart.

riskchart(predicted, actual, risks)

## End(Not run)
```

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: https://rattle.togaware.com

setupDataset *Given specific contents of env add other dataset related variables.*

Description

This rattle support function is used for encapsulating data mining objects. The supplied environment is augmented with other data derived from the supplied data, such as a sample trianing dataset, list of numeric variables, and a formula for modelling.

Usage

```
setupDataset(env, seed=NULL)
```

Arguments

env	the environment to modify.
seed	optionally set the seed for repeatability.

Details

The supplied object (an environment) is assumed to also contain the variables data (a data frame), target (a character string naming the target variable), risk (a character string naming the risk variable), and inputs (a character vector naming all the input variables). This function then adds in the variables vars (the variables used for modelling), numerics (the numeric vars within inputs), nobs (the number of observations), form (the formula for building models), train (a 70% training dataset).

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: https://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: https://rattle.togaware.com

Examples

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

`weather` *Sample dataset of daily weather observations from Canberra airport in Australia.*

Description

One year of daily weather observations collected from the Canberra airport in Australia was obtained from the Australian Commonwealth Bureau of Meteorology and processed to create this 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 recorded in millimetres). Various transformations were performed on the source data. The dataset is quite small and is useful only for repeatable demonstration of various data science operations.

The source dataset is Copyright by the Australian Commonwealth Bureau of Meteorology and is provided as part of the rattle package with permission.

Usage

```
weather
```

Format

The `weather` dataset is a data frame containing one year of daily observations from a single weather station (Canberra).

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

`Location` The common name of the location of the weather station.

`MinTemp` The minimum temperature in degrees celsius.

`MaxTemp` The maximum temperature in degrees celsius.

`Rainfall` The amount of rainfall recorded for the day in mm.

`Evaporation` The so-called Class A pan evaporation (mm) in the 24 hours to 9am.

`Sunshine` The number of hours of bright sunshine in the day.

`WindGustDir` The direction of the strongest wind gust in the 24 hours to midnight.

`WindGustSpeed` The speed (km/h) of the 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 eigths. It records how many eigths 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

The daily observations are available from https://www.bom.gov.au/climate/data. Copyright Commonwealth of Australia 2010, Bureau of Meteorology.

Definitions adapted from https://www.bom.gov.au/climate/dwo/IDCJDW0000.shtml

References

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

See Also

weatherAUS, audit.

weatherAUS *Daily weather observations from multiple Australian weather stations.*

Description

Daily weather observations from multiple locations around Australia, obtained from the Australian Commonwealth Bureau of Meteorology and processed to create this realtively large sample dataset for illustrating analytics, data mining, and data science 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 recorded in millimeters). Various transformations are performed on the data.

The `weatherAUS` dataset is regularly updated an updates of this package usually correspond to updates to this dataset. The data is updated from the Bureau of Meteorology web site.

The `locationsAUS` dataset records the location of each weather station.

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

A CSV version of this dataset is available as `https://rattle.togaware.com/weatherAUS.csv`.

Usage

```
weatherAUS
```

Format

The `weatherAUS` dataset is a data frame containing over 140,000 daily observations from over 45 Australian weather stations.

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

`Location` The common name of the location of the weather station.

`MinTemp` The minimum temperature in degrees celsius.

`MaxTemp` The maximum temperature in degrees celsius.

`Rainfall` The amount of rainfall recorded for the day in mm.

`Evaporation` The so-called Class A pan evaporation (mm) in the 24 hours to 9am.

`Sunshine` The number of hours of bright sunshine in the day.

`WindGustDir` The direction of the strongest wind gust in the 24 hours to midnight.

`WindGustSpeed` The speed (km/h) of the 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 eigths. It records how many eigths 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 https://www.bom.gov.au/climate/data. Copyright Commonwealth of Australia 2010, Bureau of Meteorology.

Definitions adapted from https://www.bom.gov.au/climate/dwo/IDCJDW0000.shtml

References

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

See Also

weather, audit.

whichNumerics *Returns a list of the names of the numeric variables in a data frame.*

Description

A rattle support function.

Usage

```
whichNumerics(data)
```

Arguments

data	a data frame.

Author(s)

<Graham.Williams@togaware.com>

References

Package home page: https://rattle.togaware.com

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 categoric variable.

The data contains no missing values and consits 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 <- "https://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* [https://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

Index