Package 'rfUtilities'

December 18, 2018

Title Random Forests Model Selection and Performance Evaluation

Version 2.1-4

Date 2018-11-17

Description Utilities for Random Forest model selection, class balance correction, significance test, cross validation and partial dependency plots.

Depends R (>= 3.3)

Imports randomForest (>= 4.6-12), cluster

Maintainer Jeffrey S. Evans <jeffrey_evans@tnc.org>

License GPL-3

NeedsCompilation no

Repository CRAN

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Author Jeffrey S. Evans [cre, aut],
Melanie A. Murphy [ctb]

Date/Publication 2018-12-17 23:50:03 UTC

R topics documented:

`accuracy` *Accuracy*

Description

Classification accuracy measures for pcc, kappa, users accuracy, producers accuracy

Usage

```
accuracy(x, y)
```

Arguments

x	vector of predicted data or table/matrix contingency table
y	vector of observed data, if x is not table/matrix contingency table

Value

A list class object with the following components:

- PCC percent correctly classified (accuracy)
- auc Area Under the ROC Curve
- users.accuracy The users accuracy
- producers.accuracy The producers accuracy
- kappa Cohen's Kappa (chance corrected accuracy)
- true.skill Hanssen-Kuiper skill score (aka true score statistic)
- sensitivity Sensitivity (aka, recall)
- specificity Specificity
- plr Positive Likelihood Ratio
- nlr Negative Likelihood Ratio
- typeI.error Type I error (omission)
- typeII.error Type II error (commission)
- gini Gini entropy index
- f.score F-score
- gain Information gain (aka precision)
- mcc Matthew's correlation
- confusion A confusion matrix

Note

- sensitivity = true positives / (true positives + false positives)
- specificity = true negatives / (true negatives + false positives)
- Type I error = 1 - specificity
- Type II error = 1 - sensitivity
- Positive Likelihood Ratio = sensitivity / (1 - specificity)
- Negative Likelihood Ratio = (1 - sensitivity) / specificity
- gain = sensitivity / ((true positives + true negatives) / n)
- auc = (tpr - fpr + 1) / 2
- F-Score = 2 * (precision * recall) / (precision + recall)
- Hanssen-Kuiper skill score (aka true score statistic) = [(tp * tn) - (fp * fn)] / [(tp + fn) + (fp + tn)], The true skill score has an expected -1 to +1, with 0 representing no discrimination.

Using the table function matrix positions for a 2x2 confusion matrix are TP(1), FN(3), FP(2), TN(4)

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

Cohen, J. (1960) A coefficient of agreement for nominal scales. Educational and Psychological Measurement 20 (1):37-46 Cohen, J. (1968) Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. Psychological Bulletin 70 (4):213-220 Powers, D.M.W., (2011). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. Journal of Machine Learning Technologies 2(1):37-63.

Examples

```
 # Two classes (vector)
 observed <- sample(c(rep("Pres",50),rep("Abs",50)), 100, replace=TRUE )
 accuracy(observed[sample(1:length(observed))], observed)

 # Two classes (contingency table)
accuracy(cbind(c(15,11), c(2,123)))

 # Multiple classes
 accuracy(iris[sample(1:150),]$Species, iris$Species)
```

`bivariate.partialDependence`

Bivariate partial-dependency plot

Description

Bivariate partial dependence provides a graphical depiction of the marginal effect of two variables on the class probability (classification) or response (regression)

Usage

```
bivariate.partialDependence(x, pred.data, v1, v2, grid.size = 20,
  which.class = 2, plot = TRUE, col.ramp = c("#ffffff", "#2a2a2a"),
  ncols = 20, ...)
```

Arguments

`x`	random forest object
`pred.data`	data.frame of independent variables used in model
`v1`	Variable 1 used in partial dependency
`v2`	Variable 2 used in partial dependency
`grid.size`	Number of grid cells (NxN) to integrate partial dependency for
`which.class`	Index of class probability (only if classification)
`plot`	(TRUE/FALSE) Plot 3D surface
`col.ramp`	Colors used in building color ramp
`ncols`	Number of colors in color ramp
`...`	Arguments passed to persp

Value

A list object with vectors of v1 (p1) and v2 (p2) and a matrix (estimate), estimate of the averaged estimates.

Note

In deriving the partial-dependence, at each plotted point, the background variables are held at their median values

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

Friedman, J.H. (2001) Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics 19(1)

Evans J.S., M.A. Murphy, Z.A. Holden, S.A. Cushman (2011). Modeling species distribution and change using Random Forests CH.8 in Predictive Modeling in Landscape Ecology eds Drew, CA, Huettmann F, Wiersma Y. Springer

Baruch-Mordo, S., J.S. Evans, J. Severson, J. D. Naugle, J. Kiesecker, J. Maestas, & M.J. Falkowski (2013) Saving sage-grouse from the trees: A proactive solution to reducing a key threat to a candidate species Biological Conservation 167:233-241

See Also

persp for persp ... plotting options

Examples

```
library(randomForest)
  data(iris)
  iris$Species <- ifelse( iris$Species == "versicolor", 1, 0 )

# Add some noise
idx1 <- which(iris$Species %in% 1)
idx0 <- which( iris$Species %in% 0)
  iris$Species[sample(idx1, 2)] <- 0
  iris$Species[sample(idx0, 2)] <- 1

# Specify model
y = iris[,"Species"]
x = iris[,1:4]

set.seed(4364)
( rf.mdl1 <- randomForest(x=x, y=factor(y)) )


( bvpd <- bivariate.partialDependence(rf.mdl1, iris,
                  v1 = "Petal.Length", v2 = "Petal.Width", shade = 0.6,
```

```
                    grid.size = 20, ncols=100, border=NA, col.ramp=c("green","blue") ) )
```

`logLoss` *Logarithmic loss (logLoss)*

Description

Evaluation of estimate quality in binomial models using cross-entropy or log likelihood loss

Usage

```
logLoss(y, p, likelihood = FALSE, global = TRUE,
  eps = 0.000000000000001)
```

Arguments

`y`	vector of observed binomial values 0,1
`p`	vector of predicted probabilities 0-1
`likelihood`	(FALSE/TRUE) return log likelihood loss, default is (FALSE) for log loss
`global`	(TRUE/FALSE) return local or global log loss values, if FALSE local values are returned
`eps`	epsilon scaling factor to avoid NaN values

Value

If likelihood TRUE the log likelihood loss will be returned. If global FALSAE, a list with observed (y), probability (p) and log loss (log.loss) otherwise, a vector of global log loss value

Note

The log loss metric, based on cross-entropy, measures the quality of predictions rather than the accuracy.

Effectively, the log loss is a measure that gauges additional error comming the estimates as opposed to the true values.

As the estimated probability diverges from its observed value the log loss increases with an expected of [0-1] where 0 would be a perferct model.

For a single sample with true value yt in 0,1 and estimated probability yp that yt = 1, the log loss is derived as: -log P(yt | yp) = -(yt log(yp) + (1 - yt) log(1 - yp))

eps is used where log loss is undefined for p=0 or p=1, so probabilities are clipped to: max(eps, min(1 - eps, p))

If likelihood is output, the eps and local arguments are ignored.

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

C.M. Bishop (2006). Pattern Recognition and Machine Learning. Springer, p. 209.

Examples

```
require(randomForest)
  data(iris)
  iris$Species <- ifelse( iris$Species == "versicolor", 1, 0 )
  # Add some noise
    idx1 <- which(iris$Species %in% 1)
    idx0 <- which( iris$Species %in% 0)
    iris$Species[sample(idx1, 2)] <- 0
    iris$Species[sample(idx0, 2)] <- 1

( mdl <- randomForest(x=iris[,1:4], y=as.factor(iris[,"Species"])) )

 # Global log loss
   logLoss(y = iris$Species, p = predict(mdl, iris[,1:4], type="prob")[,2])

 # Local log loss
   ( ll <- logLoss(y = iris$Species, p = predict(mdl, iris[,1:4],
                  type="prob")[,2], global = FALSE) )

 # Log likelihood loss
   logLoss(y = iris$Species, p = predict(mdl, iris[,1:4],
   type="prob")[,2], likelihood = TRUE)
```

`multi.collinear` *Multi-collinearity test*

Description

Test for multi-collinearity in data using qr-matrix decomposition

Usage

```
multi.collinear(x, perm = FALSE, leave.out = FALSE, n = 99,
  p = 0.0000001, na.rm = FALSE)
```

Arguments

`x`	data.frame or matrix object
`perm`	(FALSE/TRUE) Should a permutation be applied
`leave.out`	(FALSE/TRUE) Should a variable be left out at each permutation
`n`	Number of permutations
`p`	multi-collinearity threshold
`na.rm`	(FALSE/TRUE) Remove NA values

Value

If perm == TRUE a data.frame of indicating the frequency that a variable was collinear and, if leave.out = TRUE the number of times it was omitted. Otherwise, a vector of collinear variables is returned. If no colinear variables are identified a NULL is returned.

Note

A permutation approach is not available where, at each iteration, the columns are randomly rearranged and a parameter dropped. The frequency that a variable is identified as collinear is accumulated. The multi-collinearity threshold needs to be adjusted based on number of parameters. For small number(s) of variables (<20) use ~1e-07 and for larger ~0.05

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) The New S Language. Wadsworth & Brooks/Cole. Dongarra, J. J., Bunch, J. R., Moler, C. B. and Stewart, G. W. (1978) LINPACK Users Guide. Philadelphia: SIAM Publications.

Examples

```
test <- data.frame(v1=seq(0.1, 5, length=100), v2=seq(0.1, 5, length=100),
                   v3=dnorm(runif(100)), v4=dnorm(runif(100)))

# Single test
  ( cl <- multi.collinear(test) )

# Permutated test with leave out
( cl.test <- multi.collinear(test, perm = TRUE, leave.out = TRUE, n = 999) )
    cl.test[cl.test$frequency > 0,]$variables

 # Remove identified variable(s)
 head( test[,-which(names(test) %in% cl.test[cl.test$frequency > 0,]$variables)] )
```

---

occurrence.threshold *Test occurrence probability thresholds*

Description

A statistical sensitivity test for occurrence probability thresholds

Usage

```
occurrence.threshold(x, xdata, class, p = seq(0.1, 0.7, 0.02),
  type = "delta.ss")
```

Arguments

x	A classification randomForest model object
xdata	Independent data used to build model
class	What class to test
p	Vector of probability thresholds
type	What statistic to use in evaluation ("delta.ss", "sum.ss", "kappa")

Details

Available threshold evaluation statistics:

- kappa - The Kappa statistic is maximized
- sum.ss - The sum of sensitivity and specificity is maximized
- delta.ss - The absolute value of the difference between sensitivity and specificity is minimized

Value

An "occurrence.threshold" class object containing a "thresholds" vector object with evaluation statistic and probability thresholds as names.

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

Jimenez-Valverde, A., & J.M. Lobo (2007). Threshold criteria for conversion of probability of species presence to either-or presence-absence. Acta Oecologica 31(3):361-369

Liu, C., P.M. Berry, T.P. Dawson, R.G. Pearson (2005). Selecting thresholds of occurrence in the prediction of species distributions. Ecography 28:385-393.

Examples

```
library(randomForest)
 data(imports85)
  imp85 <- imports85[,-2]
  imp85 <- imp85[complete.cases(imp85), ]
  imp85[] <- lapply(imp85, function(x) if (is.factor(x)) x[, drop=TRUE] else x)

y <- ifelse( imp85$numOfDoors != "four", "0", "1")
( rf.mdl <- randomForest(y = as.factor(y), x = imp85[,-5]) )
   ( delta.ss.t <- occurrence.threshold(rf.mdl, imp85[,-5], class = "1") )
   ( sum.ss.t <- occurrence.threshold(rf.mdl, imp85[,-5], class = "1",
```

```
                                        type = "sum.ss") )
    ( kappa.ss.t <- occurrence.threshold(rf.mdl, imp85[,-5], class = "1",
                                        type = "kappa") )

par(mfrow=c(2,2))
  plot(sum.ss.t)
  plot(delta.ss.t)
  plot(kappa.ss.t)
```

`plot.occurrence.threshold` *Plot occurrence thresholds*

Description

Plot function for occurrence.threshold object

Usage

```
## S3 method for class 'occurrence.threshold'
plot(x, ...)
```

Arguments

`x`	A occurrence.threshold object
`...`	Additional arguments passed to plot

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

`plot.rf.cv` *Plot random forests cross-validation*

Description

Plot function for rf.cv object

Usage

```
## S3 method for class 'rf.cv'
plot(x, type = "cv", stat = "producers.accuracy", ...)
```

Arguments

x	A rf.cv object
type	Which result to evaluate c("cv","model")
stat	Which statistic to plot: classification: "users.accuracy", "producers.accuracy", "kappa", "oob", regression: "rmse", "mse", "var.exp", "mae", "mbe"
...	Additional arguments passed to plot

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

---

plot.rf.modelSel *Plot random forests model selection*

Description

Dot plot function for rf.modelSel importance values

Usage

```
## S3 method for class 'rf.modelSel'
plot(x, imp = "sel", ...)
```

Arguments

x	A rf.modelSel object
imp	Plot selected ("sel") or all ("all") importance used in model selection
...	Additional arguments passed to plot

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

plot.significance *Plot random forests significance*

Description

Plot function for significance object

Usage

```
## S3 method for class 'significance'
plot(x, ...)
```

Arguments

x	A significance object
...	Additional arguments passed to plot

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

print.accuracy *Print accuracy*

Description

print method for class "accuracy"

Usage

```
## S3 method for class 'accuracy'
print(x, ...)
```

Arguments

x	Object of class accuracy
...	Ignored

print.occurrence.threshold *Print occurrence.threshold*

Description

Print method for occurrence.threshold objects

Usage

```
## S3 method for class 'occurrence.threshold'
print(x, ...)
```

Arguments

x	Object of class occurrence.threshold
...	Ignored

print.rf.cv *Print random forests cross-validation*

Description

Print method for rf.cv objects

Usage

```
## S3 method for class 'rf.cv'
print(x, ...)
```

Arguments

x	Object of class rf.cv
...	Ignored

print.rf.ensembles *Print for combined random forests ensembles*

Description

print method for combined random forests ensembles

Usage

```
## S3 method for class 'rf.ensembles'
print(x, ...)
```

Arguments

x	Object of class rf.ensembles
...	Ignored

print.rf.modelSel *Print random forests model selection*

Description

Print method for rf.modelSel objects

Usage

```
## S3 method for class 'rf.modelSel'
print(x, ...)
```

Arguments

x	Object of class rf.modelSel
...	Ignored

print.significance *Print significance*

Description

print method for class "significance"

Usage

```
## S3 method for class 'significance'
print(x, ...)
```

Arguments

x	Object of class significance
...	Ignored

probability.calibration *Isotonic probability calibration*

Description

Performs an isotonic regression calibration of posterior probability to minimize log loss.

Usage

```
probability.calibration(y, p, regularization = FALSE)
```

Arguments

y	Binomial response variable used to fit model
p	Estimated probabilities from fit model
regularization	(FALSE/TRUE) should regularization be performed on the probabilities? (see notes)

Value

a vector of calibrated probabilities

Note

Isotonic calibration can correct for monotonic distortions.

regularization defines new minimum and maximum bound for the probabilities using:

pmax = (n1 + 1) / (n1 + 2), pmin = 1 / (n0 + 2); where n1 = number of prevalence values and n0 = number of null values

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

Platt, J. (1999) Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. Advances in Large Margin Classifiers (pp 61-74).

Niculescu-Mizil, A., & R. Caruana (2005) Obtaining calibrated probabilities from boosting. Proc. 21th Conference on Uncertainty in Artificial Intelligence (UAI 2005). AUAI Press.

Examples

```
library(randomForest)
  data(iris)
  iris$Species <- ifelse( iris$Species == "versicolor", 1, 0 )

  # Add some noise
  idx1 <- which(iris$Species %in% 1)
  idx0 <- which( iris$Species %in% 0)
  iris$Species[sample(idx1, 2)] <- 0
  iris$Species[sample(idx0, 2)] <- 1

# Specify model
y = iris[,"Species"]
x = iris[,1:4]
set.seed(4364)
( rf.mdl <- randomForest(x=x, y=factor(y)) )
y.hat <- predict(rf.mdl, iris[,1:4], type="prob")[,2]

# Calibrate probabilities
calibrated.y.hat <- probability.calibration(y, y.hat, regularization = TRUE)

# Plot calibrated against original probability estimate
plot(density(y.hat), col="red", xlim=c(0,1), ylab="Density", xlab="probabilities",
     main="Calibrated probabilities" )
       lines(density(calibrated.y.hat), col="blue")
         legend("topright", legend=c("original","calibrated"),
             lty = c(1,1), col=c("red","blue"))
```

---

rf.class.sensitivity *Random Forests class-level sensitivity analysis*

Description

Performs a sensitivity analysis on a specified class in a random forests model

Usage

```
rf.class.sensitivity(x, xdata, d = "1", p = 0.05, nperm = 999,
  plot = TRUE, seed = NULL, ...)
```

Arguments

x	randomForest class object
xdata	Independent variables used in model
d	Which class to perturb
p	Proportion of class to be randomized
nperm	Number of permutations
plot	Plot results (TRUE/FALSE)
seed	Random seed value
...	Additional arguments passed to randomForest

Value

List object with following components: @return mean.error Mean of RMSE @return sd.error Standard deviation of RMSE @return rmse Root mean squared error (RMSE) for each perturbed probability @return probs data.frame with "true" estimate in first column and perturbed probabilities in subsequent columns.

Note

Wildlife survey data likely decreases the proportion of imperfect detection (false absences or presences) but can still be a source of error. Because of this it is often necessary to test the model sensitivity of a given class (eg., used verses available habitat).

Model sensitivity of false absences is evaluated by randomly assigning a proportion of the specified positive class to the other, refitting the model and estimating the probabilities. Each perturbed estimate is compared against the "true" estimate. Currently only supports binomial models.

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Evans J.S., M.A. Murphy, Z.A. Holden, S.A. Cushman (2011). Modeling species distribution and change using Random Forests CH.8 in Predictive Modeling in Landscape Ecology eds Drew, CA, Huettmann F, Wiersma Y. Springer

Gardner, R.H., R.V. O'Neill, M.G. Turner, and V.H. Dale (1989). Quantifying scale-dependent effects of animal movements with simple percolation models. Landscape Ecology 3:217-227.

Examples

```
library(randomForest)
data(iris)
  y <- as.factor(ifelse(iris$Species == "setosa" |
                iris$Species == "virginica", 1, 0) )
    xdata <- iris[,1:4]

rf.mdl <- randomForest(xdata, y, ntree=501)
  ua <- rf.class.sensitivity(rf.mdl, xdata=xdata, nperm=20, ntree=501, plot=TRUE)
```

---

rf.classBalance *Random Forest Class Balance (Zero Inflation Correction) Model*

Description

Implements Evans & Cushman (2008) Random Forests class-balance (zero inflation) modeling approach.

Usage

```
rf.classBalance(ydata, xdata, p = 0.005, cbf = 3, sf = 2,
  seed = NULL, ...)
```

Arguments

ydata	Response variable using index (i.e., [,2] or [,"SPP"])
xdata	Independent variables using index (i.e., [,3:14] or [3:ncol(data)])
p	p-value of covariance convergence (do not recommend changing)
cbf	Scaling factor to test if problem is imbalanced, default is size of majority class * 3
sf	Majority subsampling factor. If sf=1 then random sample would be perfectly balanced with smallest class [s|0=n|1] whereas; sf=2 provides [s|0=(n|1*2)]
seed	Sets random seed in R global environment
...	Additional arguments passed to randomForest

Value

A rf.balanced object with the following components: @return model Final Combined Random Forests ensemble (randomForest object) @return OOB.error Out-of-bag error for each model (vector) @return confusion Confusion matrix for each model (list)

Note

This approach runs independent Random Forest models using random subsets of the majority class until covariance convergences on full data. The final model is obtained by combining independent ensembles.

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

Evans, J.S. and S.A. Cushman (2009) Gradient Modeling of Conifer Species Using Random Forest. Landscape Ecology 5:673-683.

Evans J.S., M.A. Murphy, Z.A. Holden, S.A. Cushman (2011). Modeling species distribution and change using Random Forests CH.8 in Predictive Modeling in Landscape Ecology eds Drew, CA, Huettmann F, Wiersma Y. Springer

See Also

randomForest for randomForest ... model options

Examples

```
require(randomForest)
data(iris)
  iris$Species <- as.character(iris$Species)
    iris$Species <- ifelse(iris$Species == "setosa", "virginica", iris$Species)
      iris$Species <- as.factor(iris$Species)

# Percent of "virginica" observations
length( iris$Species[iris$Species == "virginica"] ) / dim(iris)[1]*100

# Balanced model
( cb <- rf.classBalance( ydata=iris[,"Species"], xdata=iris[,1:4], cbf=1 ) )

# Calculate Kappa for each balanced model in ensemble
for(i in 1:length(cb$confusion) ) {
  print( accuracy(cb$confusion[[i]][,1:2])[5] )
}

# Evaluate cumulative and mean confusion matrix
accuracy( round((cb$confusion[[1]] + cb$confusion[[2]] + cb$confusion[[3]]))[,1:2] )
accuracy( round((cb$confusion[[1]] + cb$confusion[[2]] + cb$confusion[[3]])/3)[,1:2])
```

---

rf.combine *Combine Random Forests Ensembles*

Description

Combine two more more random forests models into a single ensemble.

Usage

```
rf.combine(...)
```

Arguments

...	two or more randomForest class objects as individual objects or a list containing models

Value

An object of class randomForest

Note

The confusion, err.rate, mse and rsq components (as well as the corresponding components in the test component, if exist) are averaged across ensembles. This is a modification of the randomForest `combine` function that returns averaged validation statistics

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

See Also

`randomForest` for randomForest details

`combine` for original combine function details

Examples

```
library(randomForest)
data(iris)

c1 <- randomForest(Species ~ ., iris, ntree=50, norm.votes=FALSE)
c2 <- randomForest(Species ~ ., iris, ntree=50, norm.votes=FALSE)
c3 <- randomForest(Species ~ ., iris, ntree=50, norm.votes=FALSE)

( class.combine <- rf.combine(c1,c2,c3) )

data(airquality)
set.seed(131)
r1 <- randomForest(Ozone ~ ., data=airquality, mtry=3,
                   importance=TRUE, na.action=na.omit)
r2 <- randomForest(Ozone ~ ., data=airquality, mtry=3,
                   importance=TRUE, na.action=na.omit)
r3 <- randomForest(Ozone ~ ., data=airquality, mtry=3,
                   importance=TRUE, na.action=na.omit)

( regress.combine <- rf.combine(r1,r2,r3) )
```

`rf.crossValidation` *Random Forest Classification or Regression Model Cross-validation*

Description

Implements a permutation test cross-validation for Random Forests models

Usage

```
rf.crossValidation(x, xdata, p = 0.1, n = 99, seed = NULL,
  normalize = FALSE, bootstrap = FALSE, trace = FALSE, ...)
```

Arguments

`x`	random forest object
`xdata`	x data used in model
`p`	Proportion data withhold (default p=0.10)
`n`	Number of cross validations (default n=99)
`seed`	Sets random seed in R global environment
`normalize`	(FALSE/TRUE) For regression, should rmse, mbe and mae be normalized using (max(y) - min(y))
`bootstrap`	(FALSE/TRUE) Should a bootstrap sampling be applied. If FALSE, an n-th percent withold will be conducted
`trace`	Print iterations
`...`	Additional arguments passed to Random Forests

Details

For classification problems, the cross-validation statistics are based on the prediction error on the withheld data: Total observed accuracy represents the percent correctly classified (aka, PCC) and is considered as a naive measure of agreement. The diagonal of the confusion matrix represents correctly classified observations where off-diagonals represent cross-classification error. The primary issue with this evaluation is that does not reveal if error was evenly distributed between classes. To represent the balance of error one can use omission and commission statistics such as estimates of users and producers accuracy. User's accuracy corresponds to error of commission (inclusion), observations being erroneously included in a given class. The commission errors are represented by row sums of the matrix. Producer's accuracy corresponds to error of omission (exclusion), observations being erroneously excluded from a given class. The omission errors are represented by column sums of the matrix. None of the previous statistics account for random agreement influencing the accuracy measure. The kappa statistic is a chance corrected metric that reflects the difference between observed agreement and agreement expected by random chance. A kappa of k=0.85 would indicate that there is 85

- pcc = [Number of correct observations / total number of observations]
- pcc = [Number of correct observations / total number of observations]

- producers accuracy = [Number of correct / total number of correct and omission errors]
- k = (observed accuracy - chance agreement) / (1 - chance agreement) where; change agreement = sum[product of row and column totals for each class]

For regression problems, a Bootstrap is constructed and the subset models MSE and percent variance explained is reported. Additional, the RMSE between the withheld response variable (y) and the predicted subset model

Value

For classification a "rf.cv"", "classification" class object with the following components:

- cross.validation$cv.users.accuracy Class-level users accuracy for the subset cross validation data
- cross.validation$cv.producers.accuracy Class-level producers accuracy for the subset cross validation data
- cross.validation$cv.oob Global and class-level OOB error for the subset cross validation data
- model$model.users.accuracy Class-level users accuracy for the model
- model$model.producers.accuracy Class-level producers accuracy for the model
- model$model.oob Global and class-level OOB error for the model

For regression a "rf.cv", "regression" class object with the following components:

- fit.var.exp Percent variance explained from specified fit model
- fit.mse Mean Squared Error from specified fit model
- y.rmse Root Mean Squared Error (observed vs. predicted) from each Bootstrap iteration (cross-validation)
- model.mse Mean Squared Error from each Bootstrapped model
- model.varExp Percent variance explained from each Bootstrapped model

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

Evans, J.S. and S.A. Cushman (2009) Gradient Modeling of Conifer Species Using Random Forest. Landscape Ecology 5:673-683.

Murphy M.A., J.S. Evans, and A.S. Storfer (2010) Quantify Bufo boreas connectivity in Yellowstone National Park with landscape genetics. Ecology 91:252-261

Evans J.S., M.A. Murphy, Z.A. Holden, S.A. Cushman (2011). Modeling species distribution and change using Random Forests CH.8 in Predictive Modeling in Landscape Ecology eds Drew, CA, Huettmann F, Wiersma Y. Springer

See Also

randomForest for randomForest ... options

Examples

```
## Not run:
library(randomForest)

# For classification
  data(iris)
    iris$Species <- as.factor(iris$Species)
      set.seed(1234)
( rf.mdl <- randomForest(iris[,1:4], iris[,"Species"], ntree=501) )
  ( rf.cv <- rf.crossValidation(rf.mdl, iris[,1:4], p=0.10, n=99, ntree=501) )

   # Plot cross validation verses model producers accuracy
   par(mfrow=c(1,2))
     plot(rf.cv, type = "cv", main = "CV producers accuracy")
     plot(rf.cv, type = "model", main = "Model producers accuracy")

   # Plot cross validation verses model oob
   par(mfrow=c(1,2))
     plot(rf.cv, type = "cv", stat = "oob", main = "CV oob error")
     plot(rf.cv, type = "model", stat = "oob", main = "Model oob error")

# For regression
data(airquality)
airquality <- na.omit(airquality)
rf.mdl <- randomForest(y=airquality[,"Ozone"], x=airquality[,2:4])
( rf.cv <- rf.crossValidation(rf.mdl, airquality[,2:4], p=0.10, n=99, ntree=501) )
 par(mfrow=c(2,2))
   plot(rf.cv)
   plot(rf.cv, stat = "mse")
   plot(rf.cv, stat = "var.exp")
plot(rf.cv, stat = "mae")

## End(Not run)
```

`rf.effectSize` *Random Forest effect size*

Description

Parameter effect size based on partial dependency (Cafri & Bailey, 2016)

Usage

```
rf.effectSize(x, y, ...)
```

Arguments

x	A randomForest model object
y	A vector represent the independent variable of intrests
...	Arguments passed to the partial dependency function, requires x.var, pred.data,

Value

A vector (single value) of the parameter effect size

Note

Effect size based on partial dependency and parameter-weighted OLS (does not support factoral or dichotomous variables) The algorithm follows: 1) Grow a forest 2) Estimate partial dependence (for a single variable). a. Create datasets for all observation in the dataset only let them take on one value for the variable of interest while keeping values of all other variables unchanged. b. Pass the dataset through each tree and average the predictions over the trees in the forest. 3) Construct a point estimate of the proposed effect size by fitting a weighted least squares model with response based on the tree-averaged predicted values obtained in Step 2, the explanatory variable corresponding to the value used to generate each tree-averaged prediction, and weight based on the frequency each value the explanatory variable takes on in the original data. 4) For confidence intervals, repeat Steps 1-3 for as many bootstrap samples as desired Modified partialPlot function uses distinct X values to construct partial dependence for non-factor variables

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

Cafri, G., B.A. Bailey (2016) Understanding Variable Effects from Black Box Prediction: Quantifying Effects in Tree Ensembles Using Partial Dependence. Journal of Data Science 14:67-96

See Also

partialPlot for ... options

randomForest for randomForest details

Examples

```
library(randomForest)
  data(airquality)
  airquality <- na.omit(airquality)
    fit.reg <- randomForest(Ozone ~ ., data=airquality)

# Parameter effect sizes
rf.effectSize(fit.reg, y = airquality$Solar.R, pred.data = airquality, x.var = Solar.R)
rf.effectSize(fit.reg, y = airquality$Wind, pred.data = airquality, x.var = Wind)
rf.effectSize(fit.reg, y = airquality$Temp, pred.data = airquality, x.var = Temp)
rf.effectSize(fit.reg, y = airquality$Month, pred.data = airquality, x.var = Month)
```

```
rf.effectSize(fit.reg, y = airquality$Day, pred.data = airquality, x.var = Day)

## Not run:
 # Bootstrap of effect size for Wind and Temp
 B = 999
 n = nrow(airquality)
 es.boot.wind <- vector()
 es.boot.temp <- vector()
   for(i in 1:B) {
     boot.samples <- airquality[sample(1:nrow(airquality), n, replace = TRUE),]
      fmla <- stats::as.formula(paste(paste("Ozone", "~", sep=""), paste(".", collapse= "")))
         fit <- randomForest(fmla, data = boot.samples)
     es.boot.wind <- append(es.boot.wind, rf.effectSize(fit, y = boot.samples$Wind,
                         pred.data = boot.samples, x.var = Wind))
     es.boot.temp <- append(es.boot.temp, rf.effectSize(fit, y = boot.samples$Temp,
                         pred.data = boot.samples,x.var = Temp))
   }
  se <- function(x) sqrt(var(x, na.rm = TRUE) / length(na.omit(x)))
    cat("Bootstrap variance for Wind:", var(es.boot.wind), "\n")
      cat("Bootstrap standard error for Wind:", se(es.boot.wind), "\n","\n")
    cat("Bootstrap variance for Temp:", var(es.boot.temp), "\n")
  cat("Bootstrap standard error for Temp:", se(es.boot.temp), "\n")

 # Confidence intervals of Bootstrap of effect size for Wind
 p=0.95
 y <- sort(es.boot.wind)
 x <- 1:length(y)
   plx <- stats::predict(stats::loess(y ~ x), se=TRUE)
   lci = plx$fit - stats::qt(p, plx$df) * plx$se.fit
   uci = plx$fit + stats::qt(p, plx$df) * plx$se.fit
       graphics::plot(x, y, type="n", main="Effect size Bootstrap CI for Wind",
      sub=paste("confidence intervals at", p))
       graphics::polygon(c(x,rev(x)), c(lci, rev(uci)), col="grey86")
       graphics::points(x, y, pch=20, cex=0.70)
       graphics::lines(x, plx[["fit"]], lty=3)

 # Confidence intervals of Bootstrap of effect size for Temp
 p=0.95
 y <- sort(es.boot.temp)
 x <- 1:length(y)
   plx <- stats::predict(stats::loess(y ~ x), se=TRUE)
   lci = plx$fit - stats::qt(p, plx$df) * plx$se.fit
   uci = plx$fit + stats::qt(p, plx$df) * plx$se.fit
       graphics::plot(x, y, type="n", main="Effect size Bootstrap CI for Temp",
      sub=paste("confidence intervals at", p))
       graphics::polygon(c(x,rev(x)), c(lci, rev(uci)), col="grey86")
       graphics::points(x, y, pch=20, cex=0.70)
       graphics::lines(x, plx[["fit"]], lty=3)

 # Plot bootstrap of wind effect size
 pdf <- density(es.boot.wind)
 plot(pdf, type="n", main="Bootstrap of effect size wind (n=99)",
      ylab="p", xlab="effect size")
```

```
  polygon(pdf, col="grey")
  abline(v=mean(es.boot.wind))
    abline(v=mean(es.boot.wind)-sd(es.boot.wind), col="blue", lty=3)
    abline(v=mean(es.boot.wind)+sd(es.boot.wind), col="blue", lty=3)

# Plot bootstrap of temp effect size
pdf <- density(es.boot.temp)
plot(pdf, type="n", main="Bootstrap of effect size temp (n=99)",
     ylab="p", xlab="effect size")
  polygon(pdf, col="grey")
  abline(v=mean(es.boot.temp))
    abline(v=mean(es.boot.temp)-sd(es.boot.temp), col="blue", lty=3)
    abline(v=mean(es.boot.temp)+sd(es.boot.temp), col="blue", lty=3)

## End(Not run)
```

`rf.imp.freq` *Random Forest variable selection frequency*

Description

Evaluates the frequency that an independent variables are selected greater-than/equal-to defined importance threshold

Usage

```
rf.imp.freq(x, p = 0.6, plot = TRUE)
```

Arguments

`x`	random forest object
`p`	Threshold of row standardized importance values
`plot`	Plot frequencies (TRUE/FALSE)

Value

A list class object with the following components: frequency: vars - names of independent variables used in model global - if a variable greater-than/equal-to importance threshold, else NA column for each class where greater-than/equal-to importance threshold, else NA var.freq - frequency a variable is selected for global and local importance >= importance threshold

importance: Standardized importance matrix from randomForest model

Note

Evaluates the number of times a variable is selected greater-than/equal-to defined threshold (p) for the global and local (class level) importances. This allow one to evaluate if a given variable is important to the overall model or specific classes.

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

Examples

```
require(randomForest)
data(iris)
iris.rf <- randomForest(Species ~ ., data=iris, importance=TRUE)
rf.imp.freq(iris.rf, p = 0.30)
```

---

rf.modelSel *Random Forest Model Selection*

Description

Implements Murphy et al., (2010) Random Forests model selection approach.

Usage

```
rf.modelSel(xdata, ydata, imp.scale = "mir", r = c(0.25, 0.5, 0.75),
  final.model = FALSE, seed = NULL, parsimony = NULL, ...)
```

Arguments

xdata	X Data for model
ydata	Y Data for model
imp.scale	Type of scaling for importance values (mir or se), default is mir
r	Vector of importance percentiles to test i.e., c(0.1, 0.2, 0.5, 0.7, 0.9)
final.model	Run final model with selected variables (TRUE/FALSE)
seed	Sets random seed in the R global environment. This is highly suggested.
parsimony	Threshold for competing model (0-1)
...	Additional arguments to pass to randomForest (e.g., ntree=1000, replace=TRUE, proximity=TRUE)

Details

If you want to run classification, make sure that y is a factor, otherwise the randomForest model runs in regression mode For classification problems the model selection criteria is: smallest OOB error, smallest maximum within class error, and fewest parameters. For regression problems, the model selection criteria is; largest

The "mir" scale option performs a row standardization and the "se" option performs normalization using the "standard errors" of the permutation-based importance measure. Both options result in a 0-1 range but, "se" sums to 1. The scaled importance measures are calculated as: mir = i/max(i) and

se = (i / se) / (sum(i) / se). The parsimony argument is the percent of allowable error surrounding competing models. For example, if there are two competing models, a selected model with 5 parameters and a competing model with 3 parameters, and parsimony = 0.05, if there is +/- 5 the fewer parameter model it will be selected at the final model.

Value

A list class object with the following components:

- "rf.final" Final selected model, if final = TRUE(randomForest model object)
- "sel.vars" Final selected variables (vector)
- "test" Validation parameters used on model selection (data.frame)
- "sel.importance" Importance values for selected model (data.frame)
- "importance" Importance values for all models (data.frame)
- "parameters" Variables used in each tested model (list)
- "s" Type of scaling used for importance

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Evans, J.S. and S.A. Cushman (2009) Gradient Modeling of Conifer Species Using Random Forest. Landscape Ecology 5:673-683.

Murphy M.A., J.S. Evans, and A.S. Storfer (2010) Quantify Bufo boreas connectivity in Yellowstone National Park with landscape genetics. Ecology 91:252-261

Evans J.S., M.A. Murphy, Z.A. Holden, S.A. Cushman (2011). Modeling species distribution and change using Random Forests CH.8 in Predictive Modeling in Landscape Ecology eds Drew, CA, Huettmann F, Wiersma Y. Springer

See Also

randomForest for randomForest ... model options

Examples

```
# Classification on iris data
require(randomForest)
data(iris)
  iris$Species <- as.factor(iris$Species)
( rf.class <- rf.modelSel(iris[,1:4], iris[,"Species"], seed=1234, imp.scale="mir") )
( rf.class <- rf.modelSel(iris[,1:4], iris[,"Species"], seed=1234, imp.scale="mir",
                          parsimony=0.03) )

   plot(rf.class)              # plot importance for selected variables
   plot(rf.class, imp = "all") # plot importance for all variables
```

```
 vars <- rf.class$selvars
 ( rf.fit <- randomForest(x=iris[,vars], y=iris[,"Species"]) )

# Regression on airquality data
data(airquality)
  airquality <- na.omit(airquality)
( rf.regress <- rf.modelSel(airquality[,2:6], airquality[,1], imp.scale="se") )
( rf.regress <- rf.modelSel(airquality[,2:6], airquality[,1], imp.scale="se", parsimony=0.03) )

   plot(rf.regress)              # plot importance for selected variables
   plot(rf.regress, imp = "all") # plot importance for all variables

# To use parameters from competing model
vars <- rf.regress$parameters[[3]]

# To use parameters from selected model
vars <- rf.regress$selvars

( rf.fit <- randomForest(x=airquality[,vars], y=airquality[,1]) )
```

`rf.partial.ci` *Random Forests regression partial dependency plot with confidence intervals*

Description

Plots the partial dependency, and associated confidence intervals, of a random forests regression model

Usage

```
rf.partial.ci(m, x, yname, xname, lci = 0.25, uci = 0.75,
  delta = FALSE)
```

Arguments

`m`	randomForest regression object
`x`	data.frame or matrix of independent variables used to build model
`yname`	Name of the dependent variable
`xname`	Name of the independent variable for calculating partial dependence
`lci`	Percentile of predictions to plot as the lower bound.
`uci`	Percentile of predictions to plot as the upper bound.
`delta`	Plot change in prediction between the independent variable and simulated values (Default = NULL)

Value

recordedplot object to recall plot

Note

depends: randomForest

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(randomForest)
data(airquality)
airquality <- na.omit(airquality)
rf.ozone <- randomForest(y=airquality[,"Ozone"], airquality[,2:ncol(airquality)])

par(mfrow=c(2,2))
  for(i in c("Solar.R", "Wind", "Temp", "Day")){
    rf.partial.ci(m=rf.ozone, x=airquality, yname="Ozone", xname=i, delta=TRUE)
  }
```

---

rf.partial.prob *Random Forest probability scaled partial dependency plots*

Description

Produces partial dependency plots with probability distribution based on scaled margin distances.

Usage

```
rf.partial.prob(x, pred.data, xname, which.class, w, prob = TRUE,
  plot = TRUE, smooth, conf = TRUE, smooth.parm = NULL,
  pts = FALSE, raw.line = FALSE, rug = FALSE, n.pt, xlab, ylab, main,
  ...)
```

Arguments

x	Object of class randomForest
pred.data	Training data.frame used for constructing the plot,
xname	Name of the variable for calculating partial dependence
which.class	The class to focus on
w	Weights to be used in averaging (if not supplied, mean is not weighted)
prob	Scale distances to probabilities

plot	(TRUE/FALSE) Plot results
smooth	c(spline, loess) Apply spline.smooth or loess to
conf	(TRUE/FALSE) Should confidence intervals be calculated for smoothing
smooth.parm	An appropriate smoothing parameter passed to loess or smooth.spline
pts	(FALSE/TRUE) Add raw points
raw.line	(FALSE/TRUE) Plot raw line (non-smoothed)
rug	Draw hash marks on plot representing deciles of x
n.pt	Number of points on the grid for evaluating partial dependence.
xlab	x-axis plot label
ylab	y-axis plot label
main	Plot label for main
...	Additional graphical parameters passed to plot

Value

A list class object with fit x,y. If smooth=c("spline","loess") y represents smoothed scaled margin distance values

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

Evans J.S., M.A. Murphy, Z.A. Holden, S.A. Cushman (2011). Modeling species distribution and change using Random Forests CH.8 in Predictive Modeling in Landscape Ecology eds Drew, CA, Huettmann F, Wiersma Y. Springer

Baruch-Mordo, S., J.S. Evans, J. Severson, J. D. Naugle, J. Kiesecker, J. Maestas, & M.J. Falkowski (2013) Saving sage-grouse from the trees: A proactive solution to reducing a key threat to a candidate species Biological Conservation 167:233-241

See Also

smooth.spline for smooth.spline details on spar smoothing argument

loess for loess details of span smoothing argument

Examples

```
require(randomForest)
  data(iris)
  iris.rf <- randomForest(iris[,1:4], iris[,5])

# plot all parameters
par(mfrow=c(2,2))
  for(i in names(iris)[1:4]) {
    rf.partial.prob(iris.rf, iris, i, "setosa", smooth="spline",
```

```
                    n.pt=70, smooth.parm = 0.5)
   }

# Plot spline and loess smoothing for one parameter, with raw points and line
par(mfrow=c(1,2))
  rf.partial.prob(x = iris.rf, pred.data = iris, xname = "Sepal.Length",
                  which.class = "setosa", smooth = "spline", smooth.parm = 0.5,
     n.pt = 70, pts = TRUE, raw.line = TRUE, rug = TRUE)

  rf.partial.prob(x = iris.rf, pred.data = iris, xname = "Sepal.Length",
                  which.class = "setosa", smooth = "loess", smooth.parm = 0.20,
     n.pt = 70, pts = TRUE, raw.line = TRUE, rug = TRUE)
```

`rf.regression.fit` *Random Forest fit statistics*

Description

Evaluates fit and overfit of random forests regression

Usage

```
rf.regression.fit(x)
```

Arguments

x	randomForest regression object

Value

A list and rf.fit class object with "fit" matrix of fit statistics and "message" indicating overfit risk.

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(randomForest)
set.seed(131)
data(airquality)
airquality <- na.omit(airquality)
( rf.aq <- randomForest(airquality[,1:3], airquality[,"Ozone"]) )
rf.regression.fit(rf.aq)
```

`rf.significance` *Random Forest model significance test*

Description

Performs significance test for classification and regression Random Forests models.

Usage

```
rf.significance(x, xdata, q = 0.99, p = 0.05, nperm = 999, ...)
```

Arguments

`x`	randomForest class object
`xdata`	Independent variables (x) used in model
`q`	Quantile threshold to test classification models
`p`	p-value to test for significance in regression models
`nperm`	Number of permutations
`...`	Additional Random Forests arguments

Value

A list class object with the following components:

For Regression problems:

RandRsquare Vector of random R-square values

Rsquare The R-square of the "true" model

Accept Is the model significant at specified p-value (TRUE/FALSE)

TestQuantile Quantile threshold used in significance plot

pValueThreshold Specified p-value

pValue p-values of randomizations

nPerm Number of permutations

For Classification problems:

RandOOB Vector of random out-of-bag (OOB) values

RandMaxError Maximum error of randomizations

test.OOB Error if the "true" model

Accept Is the model significant at specified p-value (TRUE/FALSE)

TestQuantile Quantile threshold used in significance plot

pValueThreshold Specified p-value

pValue p-values of randomizations

nPerm Number of permutations

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

Murphy M.A., J.S. Evans, and A.S. Storfer (2010) Quantify Bufo boreas connectivity in Yellowstone National Park with landscape genetics. Ecology 91:252-261

Evans J.S., M.A. Murphy, Z.A. Holden, S.A. Cushman (2011). Modeling species distribution and change using Random Forests CH.8 in Predictive Modeling in Landscape Ecology eds Drew, CA, Huettmann F, Wiersma Y. Springer

Examples

```
## Not run:
# Regression
require(randomForest)
  set.seed(1234)
    data(airquality)
    airquality <- na.omit(airquality)
 ( rf.mdl <- randomForest(x=airquality[,2:6], y=airquality[,1]) )
   ( rf.perm <- rf.significance(rf.mdl, airquality[,2:6], nperm=99, ntree=501) )

# Classification
require(randomForest)
  set.seed(1234)
    data(iris)
      iris$Species <- as.factor(iris$Species)
 ( rf.mdl <- randomForest(iris[,1:4], iris[,"Species"], ntree=501) )
   ( rf.perm <- rf.significance(rf.mdl, iris[,1:4], nperm=99, ntree=501) )

## End(Not run)
```

`rf.unsupervised` *Unsupervised Random Forests*

Description

Performs an unsupervised Random Forests for returning clustering, based on dissimilarity, and optional neighbor distance.

Usage

```
rf.unsupervised(x, n = 2, proximity = FALSE, silhouettes = FALSE,
  clara = FALSE, ...)
```

Arguments

x	A matrix/data/frame object to cluster
n	Number of clusters
proximity	(FALSE/TRUE) Return matrix of neighbor distances based on proximity
silhouettes	(FALSE/TRUE) Return adjusted silhouette values
clara	(FALSE/TRUE) Use clara partitioning, for large data
...	Additional Random Forests arguments

Value

A vector of clusters or list class object of class "unsupervised", containing the following components:

distances Scaled proximity matrix representing dissimilarity neighbor distances

k Vector of cluster labels using adjusted silhouettes

silhouette.values Adjusted silhouette cluster labels and silhouette values

Note

Clusters (k) are derived using the random forests proximity matrix, treating it as dissimilarity neighbor distances.

The clusters are identified using a Partitioning Around Medoids where negative silhouette values are assigned to the nearest neighbor.

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

Rand, W.M. (1971) Objective Criteria for the Evaluation of Clustering Methods. Journal of the American Statistical Association, 66:846-850.

Shi, T., Seligson, D., Belldegrun, A.S., Palotie, A., and Horvath, Ss (2005) Tumor Classification by Tissue Microarray Profiling: Random Forest Clustering Applied to Renal Cell Carcinoma. Modern Pathology, 18:547-557.

See Also

randomForest for ... options

pam for details on Partitioning Around Medoids (PAM)

clara for details on Clustering Large Applications (clara)

Examples

```
library(randomForest)
data(iris)
n = 4
clust.iris <- rf.unsupervised(iris[,1:4], n=n, proximity = TRUE,
                              silhouettes = TRUE)
clust.iris$k

mds <- stats:::cmdscale(clust.iris$distances, eig=TRUE, k=n)
  colnames(mds$points) <- paste("Dim", 1:n)
  mds.col <- ifelse(clust.iris$k == 1, rainbow(4)[1],
               ifelse(clust.iris$k == 2, rainbow(4)[2],
      ifelse(clust.iris$k == 3, rainbow(4)[3],
    ifelse(clust.iris$k == 4, rainbow(4)[4], NA))))
plot(mds$points[,1:2],col=mds.col, pch=20)
pairs(mds$points, col=mds.col, pch=20)
```

---

summary.accuracy *Summarizing accuracy*

Description

Summary method for class "accuracy".

Usage

```
## S3 method for class 'accuracy'
summary(object, ...)
```

Arguments

object	Object of class accuracy
...	Ignored

---

summary.occurrence.threshold *Summarizing occurrence.threshold*

Description

Summarize occurrence.threshold

Usage

```
## S3 method for class 'occurrence.threshold'
summary(object, ...)
```

Arguments

object	Object of occurrence.threshold
...	Ignored

---

summary.rf.cv *Summarizing cross-validation*

Description

Summarizing of the rf.crossValidation function

Usage

```
## S3 method for class 'rf.cv'
summary(object, ...)
```

Arguments

object	Object of class rf.cv
...	Ignored

---

summary.rf.ensembles *Summary for combined random forests ensembles*

Description

summary method for combined random forests ensembles

Usage

```
## S3 method for class 'rf.ensembles'
summary(object, ...)
```

Arguments

object	Object of class rf.ensembles
...	Ignored

summary.rf.modelSel *Summarizing random forests model selection*

Description

Summarizing of the rf.modelSel function

Usage

```
## S3 method for class 'rf.modelSel'
summary(object, ...)
```

Arguments

object	Object of class rf.modelSel
...	Ignored

summary.significance *Summarizing significance*

Description

Summarizing of a rf.significance object

Usage

```
## S3 method for class 'significance'
summary(object, ...)
```

Arguments

object	Object of class significance
...	Ignored

Index