Package 'snpRF'

February 20, 2015

Title Random Forest for SNPs to Prevent X-chromosome SNP Importance Bias

Version 0.4

Date 2014-01-20

Depends R (>= 2.5.0), stats

Suggests RColorBrewer, MASS

Author Fortran original by Leo Breiman and Adele Cutler, R port by Andy Liaw and Matthew Wiener. Modifications of randomForest v 4.6-7 for SNPs by Greg Jenkins based on a method developed by Stacey Winham, Greg Jenkins and Joanna Biernacka.

Description A modification of Breiman and Cutler's classification random forests modified for SNP (Single Nucleotide Polymorphism) data (based on randomForest v4.6-7) to prevent X-chromosome SNP variable importance bias compared to autosomal SNPs by simulating the process of X chromosome inactivation. Classification is based on a forest of trees using random subsets of SNPs and other variables as inputs.

Maintainer Greg Jenkins <jenkins.gregory@mayo.edu>

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-01-20 18:41:24

R topics documented:

classCenter 2
combine 3
getTree 4
grow 5
importance 6
margin 7
MDSplot 8
na.roughfix 9
outlier 10
plot.snpRF 11

predict.snpRF 12
snpRF 15
snpRFcv 19
snpRFexample 21
snpRFImpute 22
snpRFNews 24
treesize 24
tuneSnpRF 25
varImpPlot 26
varUsed 27

Index **29**

`classCenter` *Prototypes of groups.*

Description

Prototypes are 'representative' cases of a group of data points, given the similarity matrix among the points. They are very similar to medoids. The function is named 'classCenter' to avoid conflict with the function `prototype` in the `methods` package.

Usage

```
classCenter(x, label, prox, nNbr = min(table(label))-1)
```

Arguments

`x`	a matrix or data frame
`label`	group labels of the rows in `x`
`prox`	the proximity (or similarity) matrix, assumed to be symmetric with 1 on the diagonal and in [0, 1] off the diagonal (the order of row/column must match that of `x`)
`nNbr`	number of nearest neighbors used to find the prototypes.

Details

This version only computes one prototype per class. For each case in `x`, the `nNbr` nearest neighors are found. Then, for each class, the case that has most neighbors of that class is identified. The prototype for that class is then the medoid of these neighbors (coordinate-wise medians for numerical variables and modes for categorical variables).

This version only computes one prototype per class. In the future more prototypes may be computed (by removing the 'neighbors' used, then iterate).

Note that for use with snpRF, the X chromosome output is somewhat difficult to interpret. However, this function is still included to mimic what is included in randomForest.

Value

A data frame containing one prototype in each row.

Author(s)

Andy Liaw (copied from RandomForest package by Greg Jenkins)

See Also

snpRF, MDSplot

Examples

```
data(snpRFexample)
eg.rf <- snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
               xchrom.names=xchrom.snps.names,x.covar=covariates,
               y=phenotype,prox=TRUE)

### interpret Xchromosome output with caution ###

dimnames(xchrom.snps)[[2]]<-paste(rep(xchrom.snps.names,each=2),1:2,sep=".")
classCenter(cbind(autosome.snps,xchrom.snps,covariates), phenotype, eg.rf$prox)
```

---

combine *Combine Ensembles of Trees*

Description

Combine two or more more ensembles of trees into one.

Usage

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

Arguments

...	list of two or more objects of class snpRF, to be combined into one.

Value

An object of class snpRF.

Note

The confusion, err.rate, mse and rsq components (as well as the corresponding components in the test compnent, if exist) of the combined object will be NULL.

Author(s)

Andy Liaw, with slight modifications for use with snpRF by Greg Jenkins

See Also

snpRF, grow

Examples

```
data(snpRFexample)
rf1 <- snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
             xchrom.names=xchrom.snps.names,x.covar=covariates,
             y=phenotype,ntree=50, norm.votes=FALSE)
rf2 <- snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
             xchrom.names=xchrom.snps.names,x.covar=covariates,
             y=phenotype,ntree=50, norm.votes=FALSE)
rf3 <- snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
             xchrom.names=xchrom.snps.names,x.covar=covariates,
             y=phenotype,ntree=50, norm.votes=FALSE)

rf.all <- combine(rf1, rf2, rf3)
print(rf.all)
```

---

getTree *Extract a single tree from a forest.*

Description

This function extracts the structure of a tree from a snpRF object.

Usage

```
getTree(rfobj, k=1, labelVar=FALSE)
```

Arguments

rfobj	a snpRF object.
k	which tree to extract?
labelVar	Should better labels be used for splitting variables and predicted class?

Details

For numerical predictors, data with values of the variable less than or equal to the splitting point go to the left daughter node.

For categorical predictors, the splitting point is represented by an integer, whose binary expansion gives the identities of the categories that goes to left or right. For example, if a predictor has four categories, and the split point is 13. The binary expansion of 13 is (1, 0, 1, 1) (because $13 = 1 * 2^0 + 0 * 2^1 + 1 * 2^2 + 1 * 2^3$), so cases with categories 1, 3, or 4 in this predictor get sent to the left, and the rest to the right.

Value

A matrix (or data frame, if `labelVar=TRUE`) with six columns and number of rows equal to total number of nodes in the tree. The six columns are:

`left daughter`	the row where the left daughter node is; 0 if the node is terminal
`right daughter`	the row where the right daughter node is; 0 if the node is terminal
`split var`	which variable was used to split the node; 0 if the node is terminal
`split point`	where the best split is; see Details for categorical predictor
`status`	is the node terminal (-1) or not (1)
`prediction`	the prediction for the node; 0 if the node is not terminal

Author(s)

Andy Liaw (copied from randomForest package by Greg Jenkins)

See Also

snpRF

Examples

```
data(snpRFexample)
## Look at the third trees in the forest.
getTree(snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
              xchrom.names=xchrom.snps.names,x.covar=covariates,
              y=phenotype,ntree=10), 3, labelVar=TRUE)
```

`grow` *Add trees to an ensemble*

Description

Add additional trees to an existing ensemble of trees.

Usage

```
## S3 method for class 'snpRF'
grow(x, how.many, ...)
```

Arguments

`x`	an object of class `snpRF`, which contains a `forest` component.
`how.many`	number of trees to add to the `snpRF` object.
`...`	currently ignored.

Value

An object of class snpRF, containing how.many additional trees.

Note

The confusion, err.rate, mse and rsq components (as well as the corresponding components in the test component, if exist) of the combined object will be NULL.

Author(s)

Andy Liaw, with slight modifications for use with snpRF by Greg Jenkins

See Also

combine, snpRF

Examples

```
data(snpRFexample)
eg.rf <- snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
               xchrom.names=xchrom.snps.names,x.covar=covariates,
               y=phenotype,ntree=50, norm.votes=FALSE)
eg.rf <- grow(eg.rf, 50)
print(eg.rf)
```

---

importance *Extract variable importance measure*

Description

This is the extractor function for variable importance measures as produced by snpRF.

Usage

```
## S3 method for class 'snpRF'
importance(x, type=NULL, class=NULL, scale=TRUE, ...)
```

Arguments

x	an object of class snpRF.
type	either 1 or 2, specifying the type of importance measure (1=mean decrease in accuracy, 2=mean decrease in node impurity).
class	which class-specific measure to return.
scale	For permutation based measures, should the measures be divided their "standard errors"?
...	not used.

Details

Two importance measures are extracted using this function. The first measure is computed by permuting OOB data: For each tree, the prediction error on the out-of-bag portion of the data is recorded (error rate for classification). Then the same is done after permuting each predictor variable. The differences between the two are then averaged over all trees, and normalized by the standard deviation of the differences. If the standard deviation of the difference is equal to 0 for a variable, the division is not done (but the average is almost always equal to 0 in that case).

The second measure is the total decrease in node impurities (measured by the Gini index) from splitting on the variable, averaged over all trees.

Value

A matrix of importance measure(s), one row for each predictor variable. The column(s) are different importance measures.

See Also

snpRF, varImpPlot

Examples

```
set.seed(4543)
data(snpRFexample)

eg.rf<-snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
             xchrom.names=xchrom.snps.names,x.covar=covariates,
             y=phenotype,keep.forest=FALSE,importance=TRUE)

importance(eg.rf)
importance(eg.rf, type=1)
```

---

margin *Margins of snpRF Classifier*

Description

Compute or plot the margin of predictions from a snpRF classifier.

Usage

```
## S3 method for class 'snpRF'
margin(x, ...)
## Default S3 method:
margin(x, observed, ...)
## S3 method for class 'margin'
plot(x, sort=TRUE, ...)
```

Arguments

x	an object of class snpRF, or a matrix of predicted probabilities, one column per class and one row per observation. For the plot method, x should be an object returned by margin.
observed	the true response corresponding to the data in x.
sort	Should the data be sorted by their class labels?
...	other graphical parameters to be passed to plot.default.

Value

For margin, the *margin* of observations from the snpRF classifier (or whatever classifier that produced the predicted probability matrix given to margin). The margin of a data point is defined as the proportion of votes for the correct class minus maximum proportion of votes for the other classes. Thus under majority votes, positive margin means correct classification, and vice versa.

Author(s)

Robert Gentlemen, with slight modifications by Andy Liaw (copied from randomForest package by Greg Jenkins)

See Also

snpRF

Examples

```
set.seed(1)
data(snpRFexample)
eg.rf<-snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
             xchrom.names=xchrom.snps.names,x.covar=covariates,
             y=phenotype,keep.forest=FALSE)
plot(margin(eg.rf))
```

MDSplot *Multi-dimensional Scaling Plot of Proximity matrix from snpRF*

Description

Plot the scaling coordinates of the proximity matrix from snpRF.

Usage

```
MDSplot(rf, fac, k=2, palette=NULL, pch=20, ...)
```

Arguments

rf	an object of class snpRF that contains the proximity component.
fac	a factor that was used as response to train rf.
k	number of dimensions for the scaling coordinates.
palette	colors to use to distinguish the classes; length must be the equal to the number of levels.
pch	plotting symbols to use.
...	other graphical parameters.

Value

The output of cmdscale on 1 - rf$proximity is returned invisibly.

Note

If k > 2, pairs is used to produce the scatterplot matrix of the coordinates.

Author(s)

Robert Gentleman, with slight modifications by Andy Liaw and modifications for use with snpRF by Greg Jenkins

See Also

snpRF

Examples

```
set.seed(1)
data(snpRFexample)
eg.rf <- snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
               xchrom.names=xchrom.snps.names,x.covar=covariates,
               y=phenotype, proximity=TRUE,keep.forest=FALSE)
MDSplot(eg.rf, phenotype)
## Using different symbols for the classes:
MDSplot(eg.rf, phenotype, palette=rep(1, 2), pch=as.numeric(phenotype))
```

---

na.roughfix *Rough Imputation of Missing Values*

Description

Impute Missing Values by median/mode.

Usage

```
na.roughfix(object, ...)
```

Arguments

object	a data frame or numeric matrix.
...	further arguments special methods could require.

Value

A completed data matrix or data frame. For numeric variables, NAs are replaced with column medians. For factor variables, NAs are replaced with the most frequent levels (breaking ties at random). If object contains no NAs, it is returned unaltered.

Note

This is used as a starting point for imputing missing values by snpRF. However as noted in snpRFImpute, only the covariates can be imputed, since genetic data should be imputed using software specifically designed for SNP imputation.

Author(s)

Andy Liaw (copied from randomForest package by Greg Jenkins)

See Also

snpRFImpute, snpRF.

Examples

```
data(snpRFexample)
covar.na <- covariates
set.seed(111)
## artificially drop some data values.
for (i in 1:2) covar.na[sample(200, sample(20)), i] <- NA
eg.roughfix <- na.roughfix(covar.na)
eg.narf <- snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
                 xchrom.names=xchrom.snps.names,x.covar=eg.roughfix,
                 y=phenotype)
print(eg.narf)
```

---

outlier *Compute outlying measures*

Description

Compute outlying measures based on a proximity matrix.

Usage

```
## Default S3 method:
outlier(x, cls=NULL, ...)
## S3 method for class 'snpRF'
outlier(x, ...)
```

Arguments

x	a proximity matrix (a square matrix with 1 on the diagonal and values between 0 and 1 in the off-diagonal positions); or an object of class snpRF.
cls	the classes the rows in the proximity matrix belong to. If not given, all data are assumed to come from the same class.
...	arguments for other methods.

Value

A numeric vector containing the outlying measures. The outlying measure of a case is computed as n / sum(squared proximity), normalized by subtracting the median and divided by the MAD, within each class.

Author(s)

Unknown (copied from randomForest package by Greg Jenkins)

See Also

snpRF

Examples

```
set.seed(1)
data(snpRFexample)
eg.rf <- snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
               xchrom.names=xchrom.snps.names,x.covar=covariates,
               y=phenotype, proximity=TRUE)
plot(outlier(eg.rf), type="h",
     col=c("red", "green", "blue")[as.numeric(phenotype)])
```

---

plot.snpRF *Plot method for snpRF objects*

Description

Plot the error rates or MSE of a snpRF object

Usage

```
## S3 method for class 'snpRF'
plot(x, type="l", main=deparse(substitute(x)), ...)
```

Arguments

x	an object of class snpRF.
type	type of plot.
main	main title of the plot.
...	other graphical parameters.

Value

Invisibly, the error rates or MSE of the snpRF object. If the object has a non-null test component, then the returned object is a matrix where the first column is the out-of-bag estimate of error, and the second column is for the test set.

Note

If the x has a non-null test component, then the test set errors are also plotted.

Author(s)

Andy Liaw, with slight modifications for use with snpRF by Greg Jenkins

See Also

snpRF

Examples

```
data(snpRFexample)
plot(snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
           xchrom.names=xchrom.snps.names,x.covar=covariates,
           y=phenotype,keep.forest=TRUE), log="y")
```

---

predict.snpRF *predict method for snpRF objects*

Description

Prediction of test data using the modified random forest algorithm implemented in snpRF.

Usage

```
## S3 method for class 'snpRF'
predict(object, newdata.autosome=NULL, newdata.xchrom=NULL,
                xchrom.names=NULL, newdata.covar=NULL, type = "response",
                norm.votes = TRUE, predict.all=FALSE, proximity = FALSE,
                nodes=FALSE, cutoff, ...)
```

Arguments

`object`	an object of class `snpRF`, as that created by the function `snpRF`.
`newdata.autosome`	A matrix of autosomal markers with each column corresponding to a SNP coded as the count of a particular allele (i.e. 0,1 or 2), and each row corresponding to a sample/individual.(Note: If all newdata.* are not given, the out-of-bag prediction in `object` is returned.)
`newdata.xchrom`	A matrix of X chromosome markers, each marker coded as two adjacent columns, alleles of a marker are coded as 0 or 1 for carrying a particular allele. Although males only have one X-chromosome, their markers are coded as 2 columns as well, the second column being a duplicate of the first. Each row of this matrix corresponds to a sample/individual. This data must be phased in chromosomal order. (Note: If all newdata.* are not given, the out-of-bag prediction in `object` is returned.)
`xchrom.names`	A vector of names for markers (1 name per marker) in the newdata.xchrom matrix ordered in the same manner as markers in newdata.xchrom.
`newdata.covar`	A matrix of covariates, each column being a different covariate and each row, a sample/individual. (Note: If all newdata.* are not given, the out-of-bag prediction in `object` is returned.)
`type`	one of `response`, `prob.` or `votes`, indicating the type of output: predicted values, matrix of class probabilities, or matrix of vote counts. `class` is allowed, but automatically converted to "response", for backward compatibility.
`norm.votes`	Should the vote counts be normalized (i.e., expressed as fractions)?
`predict.all`	Should the predictions of all trees be kept?
`proximity`	Should proximity measures be computed?
`nodes`	Should the terminal node indicators (an n by ntree matrix) be return? If so, it is in the "nodes" attribute of the returned object.
`cutoff`	(Classification only) A vector of length equal to number of classes. The 'winning' class for an observation is the one with the maximum ratio of proportion of votes to cutoff. Default is taken from the `forest$cutoff` component of `object` (i.e., the setting used when running `snpRF`).
`...`	not used currently.

Value

The object returned depends on the argument `type`:

`response`	predicted classes (the classes with majority vote).
`prob`	matrix of class probabilities (one column for each class and one row for each input).
`vote`	matrix of vote counts (one column for each class and one row for each new input); either in raw counts or in fractions (if `norm.votes=TRUE`).

If `predict.all=TRUE`, then the `individual` component of the returned object is a character matrix where each column contains the predicted class by a tree in the forest.

If `proximity=TRUE`, the returned object is a list with two components: `pred` is the prediction (as described above) and `proximity` is the proximitry matrix.

If `nodes=TRUE`, the returned object has a "nodes" attribute, which is an n by ntree matrix, each column containing the node number that the cases fall in for that tree.

NOTE: Any ties are broken at random, so if this is undesirable, avoid it by using odd number `ntree` in `snpRF()`.

Author(s)

Greg Jenkins<jenkins.gregory@mayo.edu>; modification of Andy Liaw and Matthew Wiener randomForest package function predict.randomForest.R, based on original Fortran code by Leo Breiman and Adele Cutler.

References

Breiman, L. (2001), *Random Forests*, Machine Learning 45(1), 5-32.

See Also

snpRF

Examples

```
data(snpRFexample)
set.seed(111)
ind <- sample(2, nrow(autosome.snps), replace = TRUE, prob=c(0.8, 0.2))
eg.rf <- snpRF(x.autosome=autosome.snps[ind==1,],x.xchrom=xchrom.snps[ind==1,],
               xchrom.names=xchrom.snps.names,x.covar=covariates[ind==1,],
               y=phenotype[ind==1])
eg.pred <- predict(eg.rf, newdata.autosome=autosome.snps[ind==2,],
                   newdata.xchrom=xchrom.snps[ind==2,],
                   xchrom.names=xchrom.snps.names,
                   newdata.covar=covariates[ind==2,])
table(observed = phenotype[ind==2], predicted = eg.pred)
## Get prediction for all trees.
predict(eg.rf,newdata.autosome=autosome.snps[ind==2,],
        newdata.xchrom=xchrom.snps[ind==2,],
        xchrom.names=xchrom.snps.names,
```

```
        newdata.covar=covariates[ind==2,], predict.all=TRUE)
## Proximities.
predict(eg.rf,newdata.autosome=autosome.snps[ind==2,],
        newdata.xchrom=xchrom.snps[ind==2,],
        xchrom.names=xchrom.snps.names,
        newdata.covar=covariates[ind==2,], proximity=TRUE)
## Nodes matrix.
str(attr(predict(eg.rf,newdata.autosome=autosome.snps[ind==2,],
                newdata.xchrom=xchrom.snps[ind==2,],
                xchrom.names=xchrom.snps.names,
                newdata.covar=covariates[ind==2,], nodes=TRUE), "nodes"))
```

---

snpRF *Classification with Random Forest for SNP Data*

Description

snpRF implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression. It can also be used in unsupervised mode for assessing proximities among data points. This is a modified version of the randomForest function in the randomForest package addressing issues of X-chromosome SNP importance bias by simulating the process of X-inactivation.

Usage

```
snpRF(x.autosome=NULL,x.xchrom=NULL, xchrom.names=NULL, x.covar=NULL, y,
      xtest.autosome=NULL,xtest.xchrom=NULL, xtest.covar=NULL,
      ytest=NULL, ntree=500,
      mtry=floor(sqrt(sum(c(ncol(x.autosome),ncol(x.xchrom)/2,
                            ncol(x.covar))))),
      replace=TRUE, classwt=NULL, cutoff, strata,
      sampsize = if (replace) max(c(nrow(x.autosome),nrow(x.xchrom),
                                    nrow(x.covar)))
                 else ceiling(.632*max(c(nrow(x.autosome),
                                         nrow(x.xchrom),nrow(x.covar)))),
      nodesize = 1,
      maxnodes=NULL,
      importance=FALSE, localImp=FALSE,
      proximity, oob.prox=proximity,
      norm.votes=TRUE, do.trace=FALSE,
      keep.forest=!is.null(y) && (is.null(xtest.autosome) &
                                  is.null(xtest.xchrom) &
                                  is.null(xtest.covar)),
      keep.inbag=FALSE, ...)

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

Arguments

x	a snpRF object.
x.autosome	A matrix of autosomal markers with each column corresponding to a SNP coded as the count of a particular allele (i.e. 0,1 or 2), and each row corresponding to a sample/individual.
x.xchrom	A matrix of X chromosome markers, each marker coded as two adjacent columns, alleles of a marker are coded as 0 or 1 for carrying a particular allele. Although males only have one X-chromosome, their markers are coded as 2 columns as well, the second column being a duplicate of the first. Each row of this matrix corresponsponds to a sample/individual. This data must be phased in chromosomal order.
xchrom.names	A vector of names for markers (1 name per marker) in the x.xchrom matrix ordered in the same manner as markers in x.xchrom.
x.covar	A matrix of covariates, each column being a different covariate and each row, a sample/individual.
y	A response vector. Must be a factor, regression has not been implemented. If omitted, snpRF will run in unsupervised mode.
xtest.autosome	a matrix (like x.autosome) containing predictors for the test set.
xtest.xchrom	a matrix (like x.xchrom) containing predictors for the test set.
xtest.covar	a matrix (like x.covar) containing predictors for the test set.
ytest	response for the test set.
ntree	Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
mtry	Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p) where p is number of variables in: x.autosome, half of x.xchrom, and x.covar)
replace	Should sampling of cases be done with or without replacement?
classwt	Priors of the classes. Need not add up to one.
cutoff	A vector of length equal to number of classes. The 'winning' class for an observation is the one with the maximum ratio of proportion of votes to cutoff. Default is 1/k where k is the number of classes (i.e., majority vote wins).
strata	A (factor) variable that is used for stratified sampling.
sampsize	Size(s) of sample to draw. For classification, if sampsize is a vector of the length the number of strata, then sampling is stratified by strata, and the elements of sampsize indicate the numbers to be drawn from the strata.
nodesize	Minimum size of terminal nodes. Setting this number larger causes smaller trees to be grown (and thus take less time).
maxnodes	Maximum number of terminal nodes trees in the forest can have. If not given, trees are grown to the maximum possible (subject to limits by nodesize). If set larger than maximum possible, a warning is issued.
importance	Should importance of predictors be assessed?

localImp	Should casewise importance measure be computed? (Setting this to TRUE will override importance.)
proximity	Should proximity measure among the rows be calculated?
oob.prox	Should proximity be calculated only on "out-of-bag" data?
norm.votes	If TRUE (default), the final result of votes are expressed as fractions. If FALSE, raw vote counts are returned (useful for combining results from different runs). Ignored for regression.
do.trace	If set to TRUE, give a more verbose output as snpRF is run. If set to some integer, then running output is printed for every do.trace trees.
keep.forest	If set to FALSE, the forest will not be retained in the output object. If xtest is given, defaults to FALSE.
keep.inbag	Should an n by ntree matrix be returned that keeps track of which samples are "in-bag" in which trees (but not how many times, if sampling with replacement)
...	optional parameters to be passed to the low level function snpRF.

Value

An object of class snpRF, which is a list with the following components:

call	the original call to snpRF
type	classification, or unsupervised.
predicted	the predicted values of the input data based on out-of-bag samples.
importance	a matrix with nclass + 2 columns. The first nclass columns are the class-specific measures computed as mean descrease in accuracy. The nclass + 1st column is the mean descrease in accuracy over all classes. The last column is the mean decrease in Gini index.
importanceSD	The "standard errors" of the permutation-based importance measure. For classification, a p by nclass + 1 matrix corresponding to the first nclass + 1 columns of the importance matrix. For regression, a length p vector.
localImp	a p by n matrix containing the casewise importance measures, the [i,j] element of which is the importance of i-th variable on the j-th case. NULL if localImp=FALSE.
ntree	number of trees grown.
mtry	number of predictors sampled for spliting at each node.
forest	(a list that contains the entire forest; NULL if snpRF is run in unsupervised mode or if keep.forest=FALSE.
err.rate	(classification only) vector error rates of the prediction on the input data, the i-th element being the (OOB) error rate for all trees up to the i-th.
confusion	(classification only) the confusion matrix of the prediction (based on OOB data).
votes	(classification only) a matrix with one row for each input data point and one column for each class, giving the fraction or number of (OOB) 'votes' from the random forest.
oob.times	number of times cases are 'out-of-bag' (and thus used in computing OOB error estimate)

`proximity` if `proximity=TRUE` when `snpRF` is called, a matrix of proximity measures among the input (based on the frequency that pairs of data points are in the same terminal nodes).

`test` if test set is given (through the `xtest` or additionally `ytest` arguments), this component is a list which contains the corresponding `predicted`, `err.rate`, `confusion`, `votes` for the test set. If `proximity=TRUE`, there is also a component, `proximity`, which contains the proximity among the test set as well as proximity between test and training data.

Note

For details on how the trees are stored, see the help page for `getTree`.

If `xtest.*` is given, prediction of the test set is done "in place" as the trees are grown. If `ytest` is also given, and `do.trace` is set to some positive integer, then for every `do.trace` trees, the test set error is printed. Results for the test set is returned in the `test` component of the resulting `snpRF` object. For classification, the `votes` component (for training or test set data) contain the votes the cases received for the classes. If `norm.votes=TRUE`, the fraction is given, which can be taken as predicted probabilities for the classes.

The "local" (or casewise) variable importance is computed as the increase in percent of times a case is OOB and misclassified when the variable is permuted.

Author(s)

Greg Jenkins<jenkins.gregory@mayo.edu>; modification of Andy Liaw and Matthew Wiener randomForest function in the randomForest package, based on original Fortran code by Leo Breiman and Adele Cutler.

References

Breiman, L. (2001), *Random Forests*, Machine Learning 45(1), 5-32.

Breiman, L (2002), "Manual On Setting Up, Using, And Understanding Random Forests V3.1", http://oz.berkeley.edu/users/breiman/Using_random_forests_V3.1.pdf.

Jenkins, G., Biernacka J., Winham S., Random forest for genetic analysis: Integrating the X chromosome; (Abstract #1853). Presented at the 64th Annual Meeting of The American Society of Human Genetics, Date, October 21, 2014 in San Diego, CA.

See Also

`predict.snpRF`, `varImpPlot`

Examples

```
## Classification:
data(snpRFexample)
set.seed(71)
eg.rf <- snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
               xchrom.names=xchrom.snps.names,x.covar=covariates,
               y=phenotype,importance=TRUE, proximity=TRUE)
```

```
print(eg.rf)
## Look at variable importance:
round(importance(eg.rf), 2)
## Do MDS on 1 - proximity:
eg.mds <- cmdscale(1 - eg.rf$proximity, eig=TRUE)

print(eg.mds$GOF)



## Grow no more than 4 nodes per tree:
(treesize(snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
                xchrom.names=xchrom.snps.names,x.covar=covariates,
                y=phenotype, maxnodes=4, ntree=30)))
```

---

snpRFcv *Random Forest Cross-Valdidation for feature selection*

Description

This function shows the cross-validated prediction performance of models with sequentially reduced number of predictors (ranked by variable importance) via a nested cross-validation procedure.

Usage

```
snpRFcv(trainx.autosome=NULL,trainx.xchrom=NULL,trainx.covar=NULL, trainy,
        cv.fold=5, scale="log", step=0.5,
        mtry=function(p) max(1, floor(sqrt(p))), recursive=FALSE, ...)
```

Arguments

`trainx.autosome`	A matrix of autosomal markers with each column corresponding to a SNP coded as count of a particular allele (i.e. 0,1 or 2), and each row corresponding to a sample/individual.
`trainx.xchrom`	A matrix of X chromosome markers, each marker coded as two adjacent columns, alleles of a marker are coded as 0 or 1 for carrying a particular allele. Although males only have one X-chromosome, their markers are coded as 2 columns as well, the second column being a duplicate of the first. Each row of this matrix corresponds to a sample/individual. This data must be phased in chromosomal order.
`trainx.covar`	A matrix of covariates, each column being a different covariate, and each row, a sample/individual.
`trainy`	vector of response, must be a factor and have length equal to the number of rows in `trainx.*`
`cv.fold`	number of folds in the cross-validation

scale	if "log", reduce a fixed proportion (step) of variables at each step, otherwise reduce step variables at a time
step	if log=TRUE, the fraction of variables to remove at each step, else remove this many variables at a time
mtry	a function of number of remaining predictor variables to use as the mtry parameter in the snpRF call
recursive	whether variable importance is (re-)assessed at each step of variable reduction
...	other arguments passed on to snpRF

Value

A list with the following components:

list(n.var=n.var, error.cv=error.cv, predicted=cv.pred)

n.var	vector of number of variables used at each step
error.cv	corresponding vector of error rates or MSEs at each step
predicted	list of n.var components, each containing the predicted values from the cross-validation

Author(s)

Andy Liaw, with slight modifications by Greg Jenkins

References

Svetnik, V., Liaw, A., Tong, C. and Wang, T., "Application of Breiman's Random Forest to Modeling Structure-Activity Relationships of Pharmaceutical Molecules", MCS 2004, Roli, F. and Windeatt, T. (Eds.) pp. 334-343.

See Also

snpRF, importance

Examples

```
set.seed(647)
data(snpRFexample)
result <- snpRFcv(trainx.autosome=autosome.snps,trainx.xchrom=xchrom.snps,
                  trainx.covar=covariates, trainy=phenotype)
with(result, plot(n.var, error.cv, log="x", type="o", lwd=2))

## The following can take a while to run, so if you really want to try
## it, copy and paste the code into R.

## Not run:
result <- replicate(5,snpRFcv(trainx.autosome=autosome.snps,
                              trainx.xchrom=xchrom.snps,
                              trainx.covar=covariates, trainy=phenotype),
    simplify=FALSE)
```

```
error.cv <- sapply(result, "[[", "error.cv")
matplot(result[[1]]$n.var, cbind(rowMeans(error.cv), error.cv), type="l",
        lwd=c(2, rep(1, ncol(error.cv))), col=1, lty=1, log="x",
        xlab="Number of variables", ylab="CV Error")

## End(Not run)
```

---

snpRFexample *Simulated genetic data*

Description

This is simulated genetic (autosomal and X-chromosome markers) and non-genetic data (gender, age, and smoking status).

Usage

```
data(snpRFexample)
```

Format

snpRFexample contains 6 objects:

autosome.snps A matrix of 20 autosomal SNPs with each column corresponding to a SNP coded as count of a particular allele (i.e. 0,1 or 2), and each row corresponding to a subject (n=200).

xchrom.snps A matrix of 10 X chromosome SNPs, each SNP coded as two adjacent columns (20 columns in all), alleles of a marker are coded as 0 or 1 for carrying a particular allele. Altough males only have one X-chromosome, their SNPs are coded as 2 columns as well, the second column being a duplicate of the first. Each row of this matrix corresponsponds to a subject (n=200). This data is phased and in chromosomal order.

xchrom.snps.names A vector of names for markers in the xchrom.snps matrix ordered in the same manner as markers in xchrom.snps.

covariates A matrix of 2 covariates, each column being a different covariate (ie. age and smoking status (yes=1, no=0)) and each row a subject (n=200).

phenotype A vector indicating whether each subject is a case or a control.

gender A vector indicating whether each subject is male or female.

Author(s)

Greg Jenkins

Source

Data was simulated using the hapsim R package (http://CRAN.R-project.org/package=hapsim).

References

Giovanni Montana (2012). hapsim: Haplotype Data Simulation. R package version 0.3. http://CRAN.R-project.org/package=hapsim

See Also

snpRF

Examples

```
data(snpRFexample)

stopifnot(require(snpRF))
eg.rf<-snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
             xchrom.names=xchrom.snps.names,x.covar=covariates,
             y=phenotype)
print(eg.rf)
```

---

snpRFImpute *Missing Value Imputations by snpRF*

Description

Impute missing values (in covariate predictor data only) using proximity from snpRF.

Usage

```
snpRFImpute(x.autosome=NULL,x.xchrom=NULL,x.covar, y, iter=5, ntree=300, ...)
```

Arguments

x.autosome	A matrix of autosomal markers with each column corresponding to a SNP coded as count of a particular allele (i.e. 0,1 or 2), and each row corresponding to a sample/individual.(NA's not allowed).
x.xchrom	A matrix of X chromosome markers, each marker coded as two adjacent columns, alleles of a marker are coded as 0 or 1 for carrying a particular allele. Since males only have one X-chromosome, their markers are 2 columns as well, the second column being a duplicate of the first. Each row of this matrix corresponsponds to a sample/individual. This data must be phased in chromosomal order. (NA's not allowed).
x.covar	A matrix of covariates, each column being a different covariate and each row, a sample/individual, some entries with NA.
y	Response vector, must be a factor (NA's not allowed).
iter	Number of iterations to run the imputation.
ntree	Number of trees to grow in each iteration of randomForest.
...	Other arguments to be passed to snpRF.

Details

It is assumed that the genetic data (autosome and x-chromosome) would have missing values imputed in a different manner prior to using snpRF (i.e. using a program specifically intended for imputing SNP data). Thus only missing values in the covariate matrix can be imputed using data from genetic and covariate matrices.

The algorithm starts by imputing NAs using `na.roughfix`. Then `snpRF` is called with the completed data. The proximity matrix from the randomForest is used to update the imputation of the NAs. For continuous predictors, the imputed value is the weighted average of the non-missing obervations, where the weights are the proximities. For categorical predictors, the imputed value is the category with the largest average proximity. This process is iterated `iter` times.

Note: Imputation has not (yet) been implemented for the unsupervised case. Also, Breiman (2003) notes that the OOB estimate of error from randomForest tend to be optimistic when run on the data matrix with imputed values.

Value

A matrix containing the completed covariate matrix, where NAs are imputed using proximity from randomForest. The first column contains the response.

Author(s)

Andy Liaw, with slight modifications by Greg Jenkins

References

Leo Breiman (2003). Manual for Setting Up, Using, and Understanding Random Forest V4.0. http://oz.berkeley.edu/users/breiman/Using_random_forests_v4.0.pdf

See Also

`na.roughfix`.

Examples

```
data(snpRFexample)
covar.na <- covariates
set.seed(111)
## artificially drop some data values.
for (i in 1:2) covar.na[sample(200, sample(20)), i] <- NA
set.seed(222)
eg.imputed <- snpRFImpute(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
                          x.covar=covar.na, y=phenotype)

set.seed(333)
eg.rf <- snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
               xchrom.names=xchrom.snps.names,x.covar=eg.imputed,
               y=phenotype)
print(eg.rf)
```

snpRFNews *Show the NEWS file*

Description

Show the NEWS file of the snpRF package.

Usage

```
snpRFNews()
```

Value

None.

treesize *Size of trees in an ensemble*

Description

Size of trees (number of nodes) in and ensemble.

Usage

```
treesize(x, terminal=TRUE)
```

Arguments

x	an object of class snpRF, which contains a forest component.
terminal	count terminal nodes only (TRUE) or all nodes (FALSE

Value

A vector containing number of nodes for the trees in the snpRF object.

Note

The snpRF object must contain the forest component; i.e., created with snpRF(...,keep.forest=TRUE).

Author(s)

Andy Liaw, with slight modifcations for use with snpRF by Greg Jenkins

See Also

snpRF

Examples

```
data(snpRFexample)
eg.rf <- snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
               xchrom.names=xchrom.snps.names,x.covar=covariates,
               y=phenotype)
hist(treesize(eg.rf))
```

---

tuneSnpRF *Tune snpRF for the optimal mtry parameter*

Description

Starting with the default value of mtry, search for the optimal value (with respect to Out-of-Bag error estimate) of mtry for snpRF.

Usage

```
tuneSnpRF(x.autosome=NULL, x.xchrom=NULL, xchrom.names=NULL, x.covar=NULL, y,
          mtryStart, ntreeTry=50, stepFactor=2, improve=0.05, trace=TRUE,
          plot=TRUE, doBest=FALSE, ...)
```

Arguments

x.autosome	A matrix of autosomal markers with each column correspond to a SNP coded as count of a particular allele (i.e. 0,1 or 2), and each row corresponding to a sample/individual.
x.xchrom	A matrix of X chromosome markers, each marker coded as two adjacent columns, alleles of a marker are coded as 0 or 1 for carrying a particular allele. Although males only have one X-chromosome, their markers are coded as 2 columns as well, the second column being a duplicate of the first. Each row of this matrix corresponds to a sample/individual. This data must be phased in chromosomal order.
xchrom.names	A vector of names for markers (1 name per marker) in the x.xchrom matrix ordered in the same manner as markers in x.xchrom.
x.covar	A matrix of covariates, each column being a different covariate and each row, a sample/individual.
y	A response vector. Must be a factor, regression has not been implemented.
mtryStart	starting value of mtry; default is the same as in snpRF
ntreeTry	number of trees used at the tuning step
stepFactor	at each iteration, mtry is inflated (or deflated) by this value
improve	the (relative) improvement in OOB error must be by this much for the search to continue

trace	whether to print the progress of the search
plot	whether to plot the OOB error as function of mtry
doBest	whether to run a forest using the optimal mtry found
...	options to be given to snpRF

Value

If doBest=FALSE (default), it returns a matrix whose first column contains the mtry values searched, and the second column the corresponding OOB error.

If doBest=TRUE, it returns the snpRF object produced with the optimal mtry.

See Also

snpRF

Examples

```
data(snpRFexample)
eg.res <- tuneSnpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
                    xchrom.names=xchrom.snps.names,x.covar=covariates,
                    y=phenotype, stepFactor=1.5)
```

---

varImpPlot *Variable Importance Plot*

Description

Dotchart of variable importance as measured by the modified Random Forest algorithm implemented in snpRF.

Usage

```
varImpPlot(x, sort=TRUE, n.var=min(30, nrow(x$importance)),
           type=NULL, class=NULL, scale=TRUE,
           main=deparse(substitute(x)), ...)
```

Arguments

x	An object of class snpRF.
sort	Should the variables be sorted in decreasing order of importance?
n.var	How many variables to show? (Ignored if sort=FALSE.)
type, class, scale	arguments to be passed on to importance
main	plot title.
...	Other graphical parameters to be passed on to dotchart.

Value

Invisibly, the importance of the variables that were plotted.

Author(s)

Andy Liaw, with slight modifications for use with snpRF by Greg Jenkins

See Also

snpRF, importance

Examples

```
set.seed(4543)
data(snpRFexample)
eg.rf <- snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
               xchrom.names=xchrom.snps.names,x.covar=covariates,
               y=phenotype,keep.forest=FALSE, importance=TRUE)
varImpPlot(eg.rf)
```

---

varUsed *Variables used in a random forest*

Description

Find out which predictor variables are actually used in the random forest.

Usage

```
varUsed(x, by.tree=FALSE, count=TRUE)
```

Arguments

x	An object of class snpRF.
by.tree	Should the list of variables used be broken down by trees in the forest?
count	Should the frequencies that variables appear in trees be returned?

Value

If count=TRUE and by.tree=FALSE, a integer vector containing frequencies that variables are used in the forest. If by.tree=TRUE, a matrix is returned, breaking down the counts by tree (each column corresponding to one tree and each row to a variable).

If count=FALSE and by.tree=TRUE, a list of integer indices is returned giving the variables used in the trees, else if by.tree=FALSE, a vector of integer indices giving the variables used in the entire forest.

Author(s)

Andy Liaw, with slight modifications for use with snpRF by Greg Jenkins

See Also

snpRF

Examples

```
data(snpRFexample)
set.seed(17)
varUsed(snpRF(x.autosome=autosome.snps,x.xchrom=xchrom.snps,
              xchrom.names=xchrom.snps.names,x.covar=covariates,
              y=phenotype,ntree=100))
```

Index

∗Topic **NA**
na.roughfix, 9
∗Topic **classif**
classCenter, 2
combine, 3
grow, 5
importance, 6
margin, 7
MDSplot, 8
outlier, 10
plot.snpRF, 11
predict.snpRF, 12
snpRF, 15
snpRFcv, 19
snpRFImpute, 22
snpRFNews, 24
treesize, 24
tuneSnpRF, 25
varImpPlot, 26
∗Topic **datasets**
snpRFexample, 21
∗Topic **tree**
getTree, 4
importance, 6
MDSplot, 8
plot.snpRF, 11
snpRF, 15
snpRFImpute, 22
tuneSnpRF, 25
varImpPlot, 26
varUsed, 27

autosome.snps *(*snpRFexample*)*, 21

classCenter, 2
cmdscale, *9*
combine, 3, *6*
covariates *(*snpRFexample*)*, 21

dotchart, *26*

gender *(*snpRFexample*)*, 21
getTree, 4, *18*
grow, *4*, 5

importance, 6, *20*, *26, 27*

margin, 7
MDSplot, *3*, 8

na.roughfix, 9, *23*

outlier, 10

pairs, *9*
phenotype *(*snpRFexample*)*, 21
plot.margin *(*margin*)*, 7
plot.snpRF, 11
predict.snpRF, 12, *18*
print.snpRF *(*snpRF*)*, 15

snpRF, *3–14*, 15, *20*, *22, 23*, *25–28*
snpRFcv, 19
snpRFexample, 21
snpRFImpute, *10*, 22
snpRFNews, 24

treesize, 24
tuneSnpRF, 25

varImpPlot, *7*, *18*, 26
varUsed, 27

xchrom.snps *(*snpRFexample*)*, 21