Package ‘assignPOP’

July 22, 2025

Type Package

Title Population Assignment using Genetic, Non-Genetic or Integrated Data in a Machine Learning Framework

Version 1.3.0

Author Kuan-Yu (Alex) Chen [aut, cre], Elizabeth A. Marschall [aut], Michael G. Sovic [aut], Anthony C. Fries [aut], H. Lisle Gibbs [aut], Stuart A. Ludsin [aut]

Maintainer Kuan-Yu (Alex) Chen <alexkychen@gmail.com>

Description Use Monte-Carlo and K-fold cross-validation coupled with machine-learning classification algorithms to perform population assignment, with functionalities of evaluating discriminatory power of independent training samples, identifying informative loci, reducing data dimensionality for genomic data, integrating genetic and non-genetic data, and visualizing results.

URL https://github.com/alexkychen/assignPOP

Depends R (>= 2.3.2)

Imports caret, doParallel, e1071, foreach, ggplot2, MASS, parallel, randomForest, reshape2, stringr, tree, rlang,

Suggests gtable, iterators, klaR, stringi, knitr, rmarkdown, testthat

License GPL (>= 2)

RoxygenNote 7.3.1

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2024-03-13 08:30:02 UTC

Contents

accuracy.kfold 2
accuracy.MC 2
accuracy.plot 3

assign.kfold 4
assign.matrix 6
assign.MC 7
assign.X 9
check.loci 11
compile.data 12
membership.plot 12
read.Genepop 13
read.Structure 14
reduce.allele 15

Index **16**

`accuracy.kfold` *Estimate assignment accuracies of K-fold cross-validation results*

Description

This function allows you to estimate assignment accuracies of K-fold cross-validation results. The output results can be used to make assignment accuracy plots (use function accuracy.plot) and membership probability plot (use function membership.plot)

Usage

```
accuracy.kfold(dir = NULL)
```

Arguments

`dir`	A character string to specify the folder that has your K-fold cross-validation results. A slash should be included at the end (e.g., dir="YourFolderName/").

Value

This function outputs the results in a text file (a table). It can return a data frame when a returning object is specified.

`accuracy.MC` *Estimate assignment accuracies of Monte-Carlo cross-validation results*

Description

This function allows you to estimate assignment accuracies of Monte-Carlo cross-validation results. The output results can be used to make assignment accuracy plots (use function accuracy.plot).

Usage

```
accuracy.MC(dir = NULL)
```

Arguments

`dir`	A character string to specify the folder that has your Monte-Carlo cross-validation results. A slash should be included at the end (e.g., dir="YourFolderName/").

Value

This function outputs the results in a text file (a table). It can return a data frame when a returning object is specified.

`accuracy.plot` *Make a boxplot (ggplot2 style) of assignment accuracy from cross-validation results*

Description

This functions allows you to make a boxplot of assignment accuracies estimated from Monte-Carlo or K-fold cross-validation results.

Usage

```
accuracy.plot(df, pop = "all")
```

Arguments

`df`	A dataframe of your assignment accuracy results. It could be the object returned from the function accuracy.MC() or accuracy.kfold() or a data frame imported to R via other functions (e.g., read.table(...)).
`pop`	Population names (one or multiple string characters) for making the plot. By default, it uses "all", meaning overall assignment accuracies. It creates faceted plot with one population per panel, if multiple population names are given. The specified population name should match what you entered in read.genpop() earlier.

Value

This function returns a boxplot plot using the ggplot2 library. Users can modified the plot (e.g., change color, text, etc.) using functions provided by ggplot2 library.

Examples

```
Your_df <- read.table(system.file("extdata/Rate.txt", package="assignPOP"), header=TRUE)
accuracy.plot(Your_df, pop="all")
```

assign.kfold *Population assignment test using K-fold cross-validation*

Description

This function employs K-fold cross-validation for assignment tests. The results help estimate membership probabilities of every individual. It accepts genetic-only [object returned from read.genpop() or reducel.allele()], integrated [object returned from compile.data()], or non-genetic [R data frame with header] data as input, and outputs results to text files. Several built-in options are provided. See below for more details.

Usage

```
assign.kfold(
  x,
  k.fold = c(3, 4, 5),
  train.loci = c(0.1, 0.25, 0.5, 1),
  loci.sample = "fst",
  dir = NULL,
  scaled = FALSE,
  pca.method = "mixed",
  pca.PCs = "kaiser-guttman",
  pca.loadings = F,
  model = "svm",
  svm.kernel = "linear",
  svm.cost = 1,
  ntree = 50,
  processors = 999,
  multiprocess = TRUE,
  skipQ = FALSE,
  ...
)
```

Arguments

`x`	An input object which should be the object (list) returned from the function read.genpop(), reduce.allele(), or compile.data(). It could also be a data frame (with column name) returned from read.csv() or read.table() if you're analyzing non-genetic data, such as morphormetrics, chemistry data. The non-genetic data frame should have sample ID in the first column and population label in the last column.
`k.fold`	The number of groups to be divided for each population. Use a numeric vector to specify multiple sets of k-folds.
`train.loci`	The proportion (float between 0 and 1) of loci to be used as training data. Use a numeric vector to specify multiple sets of training loci. This argument will be ignored if you're analyzing non-genetic data.

`loci.sample`	Locus sampling method, "fst" or "random". If loci.sample="fst" (default) and train.loci=0.1, it means that top 10 percent of high Fst loci will be sampled as training loci. On the other hand, if loci.sample="random", then random 10 percent of loci will be sampled as training loci. This argument will be ignored if you're analyzing non-genetic data.
`dir`	A character string to specify the folder name for saving output files. A slash at the end must be included (e.g., dir="YourFolderName/"). Otherwise, the files will be saved under your working directory.
`scaled`	A logical variable (TRUE or FALSE) to specify whether to center (make mean of each feature to 0) and scale (make standard deviation of each feature to 1) the entire dataset before performing PCA and cross-validation. Default is FALSE. As genetic data has converted to numeric data between 0 and 1, to scale or not to scale the genetic data should not be critical. However, it is recommended to set scaled=TRUE when integrated data contains various scales of features.
`pca.method`	Either a character string ("mixed", "independent", or "original") or logical variable (TRUE or FALSE) to specify how to perform PCA on non-genetic data (PCA is always performed on genetic data). The character strings are used when analyzing integrated (genetic plus non-genetic) data. If using "mixed" (default), PCA is perfromed across the genetic and non-genetic data, resulting in each PC summarizing mixed variations of genetic and non-genetic data. If using "independent", PCA is independently performed on non-genetic data. Genetic PCs and non-genetic PCs are then used as new features. If using "original", original non-genetic data and genetic PCs are used as features. The logical variable is used when analyzing non-genetic data alone. If TRUE, it performs PCA on the training data and applys the loadings to the test data. Scores of training and test data will be used as new features.
`pca.PCs`	A criterion ("Kaiser-Guttman","broken-stick", or numeric) to retain number of PCs. By default, it uses Kaiser-Guttman criterion that any PC has the eigenvalue greater than 1 will be retained as the new variable/feature. Users can set an integer to specify the number of PCs to be retained.
`pca.loadings`	A logical variable (False or True) to determine whether it prints the loadings of training data to output text files. Default is False, if set True, the overall output files could be large.
`model`	A character string to specify which classifier to use for creating predictive models. The current options include "lda", "svm", "naiveBayes", "tree", and "randomForest".
`svm.kernel`	A character string to specify which kernel to be used when using "svm" classifier. Default is "linear". Other options include "polynomial", "radial", and "sigmoid". Look up R pacakge e1071 for more details about SVM, or see a guidance at https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
`svm.cost`	A number to specify the cost for "svm" method.
`ntree`	A integer to specify how many trees to build when using "randomForest" method.
`processors`	The number of processors to be used for parallel running. By default, it uses N-1 processors in your computer.
`multiprocess`	A logical variable to determine whether using multiprocess. Default is TRUE. If set FALSE, it will only use single core to run the program.

`skipQ`	A logical variable to determine whether prompting interactive dialogue when analyzing non-genetic data. If set TRUE, default data type and original values of non-genetic data will be used.
`...`	Other arguments that could be potentially used for various models

Value

You don’t need to specify a name for the returned object when using this function. It automatically outputs results in text files to your designated folder.

`assign.matrix` *Make an assignment maxtrix from cross-validation results*

Description

This function generates a pairwise assignment matrix with mean and variation of assignment accuracies estimated across all assignment tests.

Usage

```
assign.matrix(
  dir = NULL,
  train.loci = "all",
  train.inds = "all",
  k.fold = "all"
)
```

Arguments

`dir`	A character string to specify the folder that has your cross-validation assignment results.
`train.loci`	Choose your proportions of training loci used in Monte-Carlo or K-fold cross-validation. Default is "all".
`train.inds`	Choose your numbers or proportions of training individuals used in Monte-Carlo cross-validation. Default is "all".
`k.fold`	Choose the k fold values used in K-fold cross-validation. Default is "all".

Value

The function returns a matrix in R console as well as a file named "assignment_matrix.txt" in the folder.

assign.MC *Population assignment test using Monte-Carlo cross-validation*

Description

This function employs Monte-Carlo cross-validation for assignment tests. The results help evaluate if known data set has sufficient discriminatory power. It accepts genetic-only [object returned from read.genpop() or reducel.allele()], integrated [object returned from compile.data()], or non-genetic [R data frame with header] data as input, and outputs results to text files. Several built-in options are provided. See below for more details.

Usage

```
assign.MC(
  x,
  train.inds = c(0.5, 0.7, 0.9),
  train.loci = c(0.1, 0.25, 0.5, 1),
  loci.sample = "fst",
  iterations = 20,
  dir = NULL,
  scaled = FALSE,
  pca.method = "mixed",
  pca.PCs = "kaiser-guttman",
  pca.loadings = F,
  model = "svm",
  svm.kernel = "linear",
  svm.cost = 1,
  ntree = 50,
  multiprocess = TRUE,
  processors = 999,
  skipQ = FALSE,
  ...
)
```

Arguments

`x`	An input object which should be the object (list) returned from the function read.genpop(), reduce.allele(), or compile.data(). It could also be a data frame (with column name) returned from read.csv() or read.table() if you're analyzing non-genetic data, such as morphormetrics, chemistry data. The non-genetic data frame should have sample ID in the first column and population label in the last column.
`train.inds`	The number (integer greater than 1) or proportion (float between 0 and 1) of individuals (observations) from each population to be used as training data. Use a numeric vector to specify multiple sets of training individuals. No mixture of integer and float in a vector.

`train.loci`	The proportion (float between 0 and 1) of loci to be used as training data. Use a numeric vector to specify multiple sets of training loci. This argument will be ignored if you're analyzing non-genetic data.
`loci.sample`	Locus sampling method, "fst" or "random". If loci.sample="fst" (default) and train.loci=0.1, it means that top 10 percent of high Fst loci will be sampled as training loci. On the other hand, if loci.sample="random", then random 10 percent of loci will be sampled as training loci. This argument will be ignored if you're analyzing non-genetic data.
`iterations`	Resampling times (an integer) for each combination of training individuals and loci.
`dir`	A character string to specify the folder name for saving output files. A slash at the end must be included (e.g., dir="YourFolderName/"). Otherwise, the files will be saved under your working directory.
`scaled`	A logical variable (TRUE or FALSE) to specify whether to center (make mean of each feature to 0) and scale (make standard deviation of each feature to 1) the entire dataset before performing PCA and cross-validation. Default is FALSE. As genetic data has converted to numeric data between 0 and 1, to scale or not to scale the genetic data should not be critical. However, it is recommended to set scaled=TRUE when integrated data contains various scales of features.
`pca.method`	Either a character string ("mixed", "independent", or "original") or logical variable (TRUE or FALSE) to specify how to perform PCA on non-genetic data (PCA is always performed on genetic data). The character strings are used when analyzing integrated (genetic plus non-genetic) data. If using "mixed" (default), PCA is perfromed across the genetic and non-genetic data, resulting in each PC summarizing mixed variations of genetic and non-genetic data. If using "independent", PCA is independently performed on non-genetic data. Genetic PCs and non-genetic PCs are then used as new features. If using "original", original non-genetic data and genetic PCs are used as features. The logical variable is used when analyzing non-genetic data.If TRUE, it performs PCA on the training data and applys the loadings to the test data. Scores of training and test data will be used as new features.
`pca.PCs`	A criterion ("Kaiser-Guttman","broken-stick", or numeric) to retain number of PCs. By default, it uses Kaiser-Guttman criterion that any PC has the eigenvalue greater than 1 will be retained as the new variable/feature. Users can set an integer to specify the number of PCs to be retained.
`pca.loadings`	A logical variable (TRUE or FALSE) to determine whether to output the loadings of training data to text files. Default is FALSE. Just a heads-up, the output files could take some storage space, if set TRUE.
`model`	A character string to specify which classifier to use for creating predictive models. The current options include "lda", "svm", "naiveBayes", "tree", and "randomForest". Default is "svm"(support vector machine).
`svm.kernel`	A character string to specify which kernel to be used when using "svm" classifier. Default is "linear". Other options include "polynomial", "radial", and "sigmoid". Look up R pacakge e1071 for more details about SVM, or see a guidance at https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
`svm.cost`	A number to specify the cost for "svm" method.

`ntree`	A integer to specify how many trees to build when using "randomForest" method.
`multiprocess`	A logical variable to determine whether using multiprocess. Default is TRUE. If set FALSE, it will only use single core to run the program.
`processors`	The number of processors to be used for parallel running. By default, it uses N-1 processors in your computer.
`skipQ`	A logical variable to determine whether prompting interactive dialogue when analyzing non-genetic data. If set TRUE, default data type and original values of non-genetic data will be used.
`...`	Other arguments that could be potentially used for various models

Value

You don't need to specify a name for the returned object when using this function. It automatically outputs results in text files to your designated folder.

`assign.X` *Perform a population assignment test on unknown individuals using known data*

Description

This function assigns unknown individuals to possible source populations based on known individuals and genetic or non-genetic or integrated data.

Usage

```
assign.X(
  x1,
  x2,
  dir = NULL,
  common = T,
  scaled = F,
  pca.method = "mixed",
  pca.PCs = "kaiser-guttman",
  pca.loadings = F,
  model = "svm",
  svm.kernel = "linear",
  svm.cost = 1,
  ntree = 50,
  mplot = T,
  skipQ = F,
  ...
)
```

Arguments

`x1`	An input object containing data from known individuals for building predictive models. It could be a list object returned from the function read.genpop(), reduce.allele() or compile.data(). Or, it could be a data frame containing non-genetic data returned from read.csv() or read.table().
`x2`	An input object containing data from unknown individuals to be predicted. It could be a list object returned from read.genpop(), reduce.allele(), or compile.data(). Or, it could be a data frame containing non-genetic data returned from read.csv() or read.table(). The x1 and x2 should be the same type (both are either lists or data frames).
`dir`	A character string to specify the folder name for saving output files. A slash at the end must be included (e.g., dir="YourFolderName/"). Otherwise, the files will be saved under your working directory.
`common`	A logical variable (TRUE or FALSE) to specify whether exclusively using features, the name of which is in common, between known and unknown data sets. Default is TRUE. If it is FALSE, it will stop performing analysis when inconsistent feature names were found.
`scaled`	A logical variable (TRUE or FALSE) to specify whether to center (make mean of each feature to 0) and scale (make standard deviation of each feature to 1) the dataset before performing PCA and cross-validation. Default is FALSE. As genetic data has converted to numeric data between 0 and 1, to scale or not to scale the genetic data should not be critical. However, it is recommended to set scaled=TRUE when integrated data contains various scales of features.
`pca.method`	Either a character string ("mixed", "independent", or "original") or logical variable (TRUE or FALSE) to specify how to perform PCA on non-genetic data (PCA is always performed on genetic data). The character strings are used when analyzing integrated (genetic plus non-genetic) data. If using "mixed" (default), PCA is perfromed across the genetic and non-genetic data, resulting in each PC summarizing mixed variations of genetic and non-genetic data. If using "independent", PCA is independently performed on non-genetic data. Genetic PCs and non-genetic PCs are then used as new features. If using "original", original non-genetic data and genetic PCs are used as features. The logical variable is used when analyzing non-genetic data.If TRUE, it performs PCA on the training data and applys the loadings to the test data. Scores of training and test data will be used as new features.
`pca.PCs`	A criterion ("Kaiser-Guttman","broken-stick", or numeric) to retain number of PCs. By default, it uses Kaiser-Guttman criterion that any PC has the eigenvalue greater than 1 will be retained as the new variable/feature. Users can set an integer to specify the number of PCs to be retained.
`pca.loadings`	A logical variable (TRUE or FALSE) to determine whether to output the loadings of training data to text files. Default is FALSE. Just a heads-up, the output files could take some storage space, if set TRUE.
`model`	A character string to specify which classifier to use for creating predictive models. The current options include "lda", "svm", "naiveBayes", "tree", and "randomForest". Default is "svm"(support vector machine).

`svm.kernel`	A character string to specify which kernel to be used when using "svm" classifier. Default is "linear". Other options include "polynomial", "radial", and "sigmoid". Look up R pacakge e1071 for more details about SVM, or see a guidance at https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
`svm.cost`	A number to specify the cost for "svm" method.
`ntree`	A integer to specify how many trees to build when using "randomForest" method.
`mplot`	A logical variable (TRUE or FALSE) to specify whether making a membership probability plot right after the assignment test is done. Default is TRUE.
`skipQ`	A logical variable (TRUE or FALSE) to skip data type checking on non-genetic data. Default is FALSE and will prompt questions to confirm data type. If it is TRUE, it will skip the confirmation and use data type by default (integer and float will be numeric data).
`...`	Other arguments that could be potentially used for various models

Value

This function outputs assignment results and other analytical information in text files that will be saved under your designated folder. It also outputs a membership probability plot, if permitted.

`check.loci` *Check which loci frequently have high Fst across training sets*

Description

This function reads through training locus file for each assignment test and counts the frequency of those loci and outputs the results in a text file.

Usage

```
check.loci(dir = NULL, top.loci = 20)
```

Arguments

`dir`	A character string to specify the folder with your cross-validation results. A slash should be entered at the end.
`top.loci`	An integer to specify how many top informative loci to output.

Value

This function output the results in a text file. It includes the top N informative loci in N rows, and each row has a list of loci sorted by its occurrence.

`compile.data` *Compile genetic and other non-genetic data*

Description

This function allows you to combine genetic and other non-genetic data, such as morphometrics, of the observations for assignment tests.

Usage

```
compile.data(x, add.x, method = "common", skipQ = F)
```

Arguments

`x`	A returned object (list) from the function read.genpop() or reduce.allele().
`add.x`	A file containing non-genetic data that has sample ID in the first column. The sample ID must be the same as your GENEPOP file.
`method`	A method to match sample ID between genetic and non-genetic data. The "common" method only concatenate the data that has sample ID in both files. If an individual only exists in one of the files, this individual will be discarded.
`skipQ`	A logical variable to determine whether prompting interactive dialogue. If set TRUE, input data type will be recognized as default type and not be verified by the user.

Value

This function returns a new object (list) that comprises 5 items. [[1]] data matrix including genetic and non-genetic data, [[2]] a sample ID vector, [[3]] a locus name vector, [[4]] a vector of non-genetic variable names, and [[5]] the number of non-genetic variables.

`membership.plot` *Make a membership probability plot using results from K-fold cross-validation (ggplot2 style)*

Description

This function allows you to make a membership probability plot (stacked-bar plot) using results estimated from K-fold cross-validation.

Usage

```
membership.plot(
  dir = NULL,
  style = NULL,
  non.genetic = FALSE,
  plot.k = NULL,
  plot.loci = NULL
)
```

Arguments

`dir`	A character string to specify the folder that has your K-fold cross-validation assignment results. A slash should be entered at the end.
`style`	An option for output style. If style=1, it creates the plot which individuals on the x-axis are in random order. If style=2, individuals are sorted by probabilities within each population. If style=3, individuals of different folds are in seperate plots. If style=4, individuals are separated by fold and sorted by probability.
`non.genetic`	A logical variable to specify if data are non-genetic. Set it TRUE if you're analyzing non-genetic alone.
`plot.k`	A number to specify which K of the data set should be plotted. If not given, it will prompt the question.
`plot.loci`	The proportion of training loci used in your K-fold cross-validation analysis. Specify one of the numbers here to skip question prompt.

Value

This function returns a stacked-bar plot using the ggplot2 library. Users can modified the plot (e.g., change color, text, etc.) using functions provided by ggplot2 library.

`read.Genepop` *Read GENEPOP format file*

Description

This function allows you to import a GENEPOP format file into R. Population names can be specified in the argument. See http://genepop.curtin.edu.au/help_input.html for details about GENEPOP format.

Usage

```
read.Genepop(x, pop.names = NULL, haploid = FALSE, pos = 1)
```

Arguments

`x`	GENEPOP file or path to the file. The filename extension (e.g., .txt) should be included.
`pop.names`	A character string vector for population names. The order of the name should be the same with the order (top to down) in your GENEPOP file.
`haploid`	A logical variable (TRUE or FALSE) to specify whether your dataset is haploid data. Default is FALSE.
`pos`	A parameter for program development use; users can ignore it.

Value

This function returns a list comprising three elements. 1. YOU_NAME_IT$DataMatrix: A matrix of genetic data with a population name label ($popNameVector) in the last column. 2. YOU_NAME_IT$SampleID: A vector of sample ID. 3. YOU_NAME_IT$LocusName: A vector of locus name.

References

Rousset, F. 2008. Genepop'007: a complete reimplementation of the Genepop software for Windows and Linux. Mol. Ecol. Resources 8: 103-106

Examples

```
# infile <- read.Genepop("Your_Genepop_File.txt", pop.names=c("pop_A", "pop_B", "pop_C"))
```

`read.Structure` *Read Structure format file*

Description

This function allows you to import a STRUCTURE format file into R. The first row should be locus name (either with or withour column names for sample ID and population label); the first column should be sample ID; the second column should be population label; the rest are genotype. Use "-9" for missing alleles.

Usage

```
read.Structure(x, ploidy = 2)
```

Arguments

`x`	STRUCTURE file or path to the file. The filename extension (e.g., .txt) should be included.
`ploidy`	An integer of 1, 2, 3, or 4, to indicate haploid, diploid, triploid, or tetraploid data. Default is 2 (diploid).

Value

This function returns a list comprising three elements. 1. YOU_NAME_IT$DataMatrix: A matrix of genetic data with a population name label ($popNameVector) in the last column. 2. YOU_NAME_IT$SampleID: A vector of sample ID. 3. YOU_NAME_IT$LocusName: A vector of locus name.

References

Pritchard, J.K., Stephens, M. and Donnelly, P., 2000. Inference of population structure using multi-locus genotype data. Genetics, 155(2), pp.945-959.

Examples

```
# infile <- read.Structure("Your_Structure_File.txt")
```

`reduce.allele` *Remove low variance alleles (dimensionality reduction)*

Description

This function helps remove alleles that have low variance in the data set such that it can speed up further analyses for a large data set (e.g., > 10K SNPs).

Usage

```
reduce.allele(x, p = 0.95)
```

Arguments

x	A returned object (a list) from the function read.genpop().
p	A threshold of variance for the alleles to be removed. For example, if p = 0.95 (default setting), an allele occupied more than 95 percents across all the samples will be removed.

Value

This function return the same object as the function read.genpop() except that the number of columns in the matrix [[1]] is reduced and so is the locus name [[3]].

Index

accuracy.kfold, 2
accuracy.MC, 2
accuracy.plot, 3
assign.kfold, 4
assign.matrix, 6
assign.MC, 7
assign.X, 9

check.loci, 11
compile.data, 12

membership.plot, 12

read.Genepop, 13
read.Structure, 14
reduce.allele, 15