Package 'allelematch'

July 24, 2026

Type Package

Title Identifying Unique Multilocus Genotypes where Genotyping Error and Missing Data may be Present

Version 3.0.0

Date 2026-07-21

Author Paul Galpern [aut],
Micheline Manseau [aut],
Pete Hettinga [aut],
Karen Smith [aut],
Paul Wilson [aut],
Todd Cross [cre]

Maintainer Todd Cross <todd.cross@gmail.com>

Description Tools for the identification of unique multilocus genotypes when both genotyping error and missing data may be present. Includes a data pre-screening utility to analyze pairwise locus overlap and protect against underlying mathematical sorting constraints. Targeted for use with large datasets and databases containing multiple samples of each individual (a common situation in conservation genetics, particularly in non-invasive wildlife sampling applications). Functions explicitly incorporate missing data and can tolerate allele mismatches created by genotyping error. If you use this package, please cite the original publication in Molecular Ecology Resources (Galpern et al., 2012), the details for which can be generated using citation('allelematch'). The complete user manual and analytical tutorials are included locally as an R vignette and can be accessed within an active R session using vignette('allelematch').

License GPL (>= 2)

Depends R (>= 3.5.0), dynamicTreeCut

URL <doi:10.1111%2Fj.1755-0998.2012.03137.x>

LazyLoad yes

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2026-07-24 04:30:02 UTC

Contents

allelematch-package 2
amAlleleFreq 3
amCluster 5
amCSSForHTML 10
amDataset 11
amExampleData 13
amMatrix 14
amPairwise 16
amPreCheck 19
amUnique 20
amUniqueProfile 25

Index **29**

`allelematch-package` *Identification of unique multilocus genotypes*

Description

Tools for the identification of unique multilocus genotypes when both genotyping error and missing data may be present; targeted for use with large datasets and databases containing multiple samples of each individual (a common situation in conservation genetics, particularly in non-invasive wildlife sampling applications). Functions explicitly incorporate missing data and can tolerate allele mismatches created by genotyping error.

Details

Package:	allelematch
Type:	Package
Version:	3.0.0
Date:	2026-07-23
License:	GPL (>= 2)
Requires:	dynamicTreeCut
LazyLoad:	yes

Core Functions:

- `amDataset`: Produces an input dataset object for allelematch routines.
- `amPreCheck`: Pre-screens an amDataset object for severe missing data loads to prevent runtime sorting failures.

- amMatrix: Produces a dissimilarity matrix.
- amPairwise: Performs pairwise matching of genotypes.
- amCluster: Clusters genotypes using dynamic tree cutting.
- amAlleleFreq: Produces allele frequencies from an amDataset object.
- amUnique: Identifies unique genotypes.
- amUniqueProfile: Finds optimal parameters for amUnique.

The complete user manual and analytical tutorials describing the operation of the software in detail are included locally as a package vignette. It is installed automatically with the package and can be accessed within an active R session using `vignette("allelematch")`.

Simulations examining the performance of these tools have also been performed, and results are available in the publication associated with this package. Please refer to the publication:

Galpern P, Manseau, M, Hettinga P, Smith K, and Wilson P. (2012) allelematch: an R package for identifying unique multilocus genotypes where genotype error and missing data may be present. *Molecular Ecology Resources* 12:771-778.

Use `citation("allelematch")` for the full citation. Please also use this publication when citing the package.

An important core element of the package is dynamic tree cutting, and this is made possible via the cutreeHybrid function within the dynamicTreeCut package for R (Langfelder et al., 2008).

Author(s)

Paul Galpern (<pgalpern@gmail.com>), Todd Cross (<todd.cross@gmail.com>)

`amAlleleFreq` *Determine allele frequencies*

Description

Determines the allele frequencies for each locus in a multilocus genetic dataset by first removing missing observations. It requires an `amDataset` object and a map vector relating each column of the dataset to a genetic locus.

Usage

```
amAlleleFreq(
amDatasetFocal,
multilocusMap = NULL
)

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

Arguments

amDatasetFocal	An amDataset object
multilocusMap	Optionally, a vector of integers or strings giving the mappings onto loci for all genotype columns in amDatasetFocal. When omitted, columns are assumed to be paired (i.e., diploid loci with alleles in adjacent columns). See details.
x	An amDataset object.
...	Additional arguments to summary

Details

This function is called by amUnique.

multilocusMap is often not required, as amDataset objects will typically consist of paired columns of genotypes, where each pair is a separate locus. In cases where this is not the case (e.g., gender is given in only one column), a map vector must be specified.

Example: amDataset consists of gender followed by 4 diploid loci in paired columns
multilocusMap = c(1, 2, 2, 3, 3, 4, 4, 5, 5)
or equally
multilocusMap=c("GENDER", "LOC1", "LOC1", "LOC2", "LOC2", "LOC3", "LOC4","LOC4")

Value

An amAlleleFreq object

Author(s)

Paul Galpern (<pgalpern@gmail.com>)

References

For more information and for a complete vignette, please access via the Data S1 Supplementary documentation and tutorials (PDF) located at <doi:10.1111/j.1755-0998.2012.03137.x>.

See Also

amUnique

Examples

```
## Not run:
data("amExample5")

## Produce amDataset object
myDataset1 <-
amDataset(
```

```
amExample5,
missingCode = "-99",
indexColumn = 1,
metaDataColumn = 2
)

## Usage
myAlleleFreq <-
amAlleleFreq(
myDataset1,
multilocusMap = c(1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 10,
11, 11)
)

## Produce amDataset object, but remove gender column
myDataset2 <-
amDataset(
amExample5,
missingCode = "-99",
indexColumn = 1,
metaDataColumn = 2,
ignoreColumn = "gender"
)

## Because all columns are paired, usage is simpler
myAlleleFreq <- amAlleleFreq(myDataset2)

## End(Not run)
```

---

amCluster *Clustering of multilocus genotypes*

Description

Performs clustering of multilocus genotypes to identify unique consensus and singleton genotypes and generates analysis output in formatted text, HTML, or CSV. These functions are usually called by amUnique. This interface remains to enable a better understanding of how amUnique operates. For more information see example.

There are three steps to this analysis: (1) identify the dissimilarity between pairs of genotypes using a metric which takes missing data into account, (2) cluster this dissimilarity matrix using a standard hierarchical agglomerative clustering approach, and (3) use a dynamic tree cutting approach to identify clusters.

Usage

```
amCluster(
amDatasetFocal,
runUntilSingletons = TRUE,
```

```
cutHeight = 0.3,
missingMethod = 2,
consensusMethod = 1,
clusterMethod = "complete"
)

amHTML.amCluster(
x,
htmlFile = NULL,
htmlCSS = amCSSForHTML()
)

amCSV.amCluster(
x,
csvFile
)

  ## S3 method for class 'amCluster'
summary(
    object,
html = NULL,
csv = NULL,
...
)
```

Arguments

amDatasetFocal	An amDataset object containing genotypes to cluster.
runUntilSingletons	When runUntilSingletons = TRUE, the analysis runs recursively with the unique individuals determined in one analysis feeding into the next until no more clusters are formed; applicable when the goal is to thin a dataset to unique genotypes. For more manual control over the process, use runUntilSingletons = FALSE. See details and examples.
cutHeight	Sets the tree cutting height using the hybrid method in cutreeHybrid. See details and cutreeHybrid for more information.
missingMethod	The method used to determine the similarity of multilocus genotypes when data is missing. The default, (missingMethod = 2), is preferable in all cases. See amMatrix.
consensusMethod	The method (an integer) used to determine the consensus multilocus genotype from a cluster of multilocus genotypes. See details.
clusterMethod	The method used by hclust for clustering. Only the default clusterMethod = "complete" performs acceptably in simulations. This option remains for experimental reasons.

object, x	An amPairwise object.
htmlFile	HTML filepath to create. If htmlFile = NULL, a file is created in the operating system temporary directory and is then opened in the default browser.
htmlCSS	String containing a valid cascading style sheet. A default style sheet is provided in amCSSForHTML. See amCSSForHTML for details of how to tweak this CSS.
html	If html = NULL or html=FALSE, formatted textual output is displayed on the console. If html = TRUE, the summary.amCluster method produces and loads an HTML file in the default browser. html can also contain a path to a file where HTML output will be written.
csvFile, csv	CSV filepath to create containing only the unique genotypes determined in the clustering.
...	Additional arguments to summary.amCluster

Details

Selecting an appropriate cutHeight parameter (also known as the d-hat criterion) is essential. Typically this function is called from amUnique, and the conversion between alleleMismatch (m-hat) and cutHeight (d-hat) will be done automatically. Selecting an appropriate value for alleleMismatch (m-hat) can be done using amUniqueProfile. See the supplementary documentation for an explanation of how these parameters are related.

runUntilSingletons=TRUE provides an efficient and reliable way to determine the unique individuals in a dataset if the dataset meets certain criteria. To understand how the clustering is thinning the dataset run this recursion manually using runUntilSingletons=FALSE. An example is provided below.

cutHeight in practice gives the amount of dissimilarity (using the metric described in amMatrix) required for two multilocus genotypes to be declared different (also known as d-hat). The default setting for consensusMethod performs well.

consensusMethod	
1	Genotype with max similarity to others in the cluster is consensus (DEFAULT)
2	Genotype with max similarity to others in the cluster is consensus then interpolate missing alleles using n
3	Genotype with min missing data is consensus
4	Genotype with min missing data is consensus then interpolate missing alleles using mode non-missing all

Value

amCluster object or side effects: analysis summary written to an HTML file or to the console, or written to a CSV file.

Note

There is an additional side effect of `html = TRUE` (or of `htmlFile = NULL`). If required, there is a clean up of the operating system temporary directory where AlleleMatch temporary HTML files are stored. Files that match the pattern am*.html and are older than 24 hours are deleted from this temporary directory.

Author(s)

Paul Galpern (<pgalpern@gmail.com>)

References

For a complete vignette, please access via the Data S1 Supplementary documentation and tutorials (PDF) located at <doi:10.1111/j.1755-0998.2012.03137.x>.

See Also

amDataset, amMatrix, amPairwise, amUnique, amUniqueProfile

Examples

```
## Not run:
data("amExample5")

## Produce amDataset object
myDataset <-
amDataset(
amExample5,
missingCode = "-99",
indexColumn = 1,
metaDataColumn = 2,
ignoreColumn = "gender"
)

## Usage
myCluster <-
amCluster(
myDataset,
cutHeight = 0.2
)

## Display analysis as HTML in default browser
summary.amCluster(
myCluster,
html = TRUE
)

## Save analysis to HTML file
summary.amCluster(
myCluster,
html = "myCluster.htm"
```

```
)

## Display analysis as formatted text on the console
summary.amCluster(myCluster)

## Save unique genotypes only to a CSV file
summary.amCluster(
myCluster,
csv = "myCluster.csv"
)

## Demonstration of how amCluster operates
## Manual control over the recursion in amCluster()
summary.amCluster(
myCluster1 <-
amCluster(
myDataset,
runUntilSingletons = FALSE,
cutHeight = 0.2
),
html = TRUE
)
summary.amCluster(
myCluster2 <-
amCluster(
myCluster1$unique,
runUntilSingletons = FALSE,
cutHeight = 0.2
),
html = TRUE
)
summary.amCluster(
myCluster3 <-
amCluster(
myCluster2$unique,
runUntilSingletons = FALSE,
cutHeight = 0.2
),
html = TRUE
)
summary.amCluster(
myCluster4 <-
amCluster(
myCluster3$unique,
runUntilSingletons = FALSE,
cutHeight = 0.2
),
html = TRUE
)
## No more clusters, therefore stop.

## End(Not run)
```

amCSSForHTML *Produce cascading style sheet (CSS) for HTML*

Description

Returns a string containing the CSS code for embedding in HTML output by amHTML.amPairwise and amHTML.amUnique.

Usage

```
amCSSForHTML()
```

Details

This function is used internally. It can also be used as a basis to tweak the CSS code if different output formatting and colour-coding are desired. See examples.

Value

A string containing a complete cascading style sheet.

Author(s)

Paul Galpern (<pgalpern@gmail.com>)

References

For more information and for a complete vignette, please access via the Data S1 Supplementary documentation and tutorials (PDF) located at <doi:10.1111/j.1755-0998.2012.03137.x>.

See Also

amHTML.amPairwise, amHTML.amUnique

Examples

```
## Not run:
data("amExample5")

## Produce CSS file for editing
cat(amCSSForHTML(), file = "myCSS.css")

## Edit myCSS.css manually using text editor, then reimport as follows
myCSS <-
paste(
readLines("myCSS.css"),
collapse = "\n"
)
```

```
## Perform an allelematch unique analysis
myUnique <-
amUnique(
amDataset(
amExample5,
missingCode = "-99",
indexColumn = 1,
metaDataColumn = 2,
ignoreColumn = "gender"
),
alleleMismatch = 3
)

## Produce HTML output using tweaked CSS
amHTML.amUnique(
myUnique,
html = "myUnique.htm",
htmlCSS = myCSS
)

## End(Not run)
```

`amDataset` *Prepare a dataset for use with allelematch*

Description

Given an input matrix or `data.frame` produce a amDataset object suitable for use with other allelematch functions.

Usage

```
amDataset(
multilocusDataset,
missingCode = "-99",
indexColumn = NULL,
metaDataColumn = NULL,
ignoreColumn = NULL
)

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

Arguments

`multilocusDataset`
: A `matrix` or `data.frame` containing samples in rows and alleles in columns. Sampling IDs and meta-data may be specified in up to two additional columns.

missingCode	A character string giving the code used for missing data. Missing data may also be represented as NA.
indexColumn	Optional. A character string giving the column name, or an integer giving the column number containing the sampling ID or index information. If an index is not supplied, then the function creates an alphabetical index.
metaDataColumn	Optional. A character string giving the column name, or an integer giving the column number containing the meta-data.
ignoreColumn	Optional. A vector of character string(s) giving the column name(s) or integer(s) giving the column number(s) that should be removed from the input dataset (i.e., those that matching and clustering should not consider).
x	An amDataset object.
...	Additional arguments to summary.

Details

Examine amExampleData for an example of a typical input dataset in the diploid case. (Typically these files will be the CSV output from allele calling software). Sample index or ID information and sample meta-data may be specified in two additional columns. Columns can optionally be given names, and these are carried through analyses. If column names are not given, appropriate names are produced.

Each datum is treated as a character string in allelematch functions, enabling the mixing of numeric and alphanumeric data.

The multilocus dataset can contain any number of diploid or haploid markers, and these can be in any order. Thus in the diploid case there should be two columns for each locus (named, say, locus1a and locus1b). Please note that AlleleMatch functions pay no attention to genetics. In other words, each column is considered a comparable state. Thus matching and clustering of multilocus genotypes is done on the basis of superficial similarity of the data matrix rows, rather than on any appreciation of the allelic states at each locus. See amPairwise for more discussion.

For this reason it is important when working with diploid data to ensure that identical individuals will have identical alleles in each column. This can be achieved by sorting each locus so that in each case the lower length allele appears in, say, a column "locus1a" and the higher in column "locus1b." This pattern is likely the default in allele calling software and sorting will typically not be required unless data are derived from an unusual source.

Only one meta-data column is possible with allelematch. If multiple columns must be associated with a given sample for downstream analyses, try pasting them together into one string with an appropriate separator, and separating them later when allelematch analyses are concluded.

Value

An amDataset object.

Author(s)

Paul Galpern (<pgalpern@gmail.com>)

References

For a complete vignette, please access via the Data S1 Supplementary documentation and tutorials (PDF) located at <doi:10.1111/j.1755-0998.2012.03137.x>.

See Also

amPairwise, amUnique, amExampleData

Examples

```
## Not run:
data("amExample5")

## Typical usage
myDataset <-
amDataset(
amExample5,
missingCode = "-99",
indexColumn = 1,
metaDataColumn = 2,
ignoreColumn = "gender"
)

## Access elements of amDataset object
myMetaData <- myDataset$metaData
mySamplingID <- myDataset$index
myAlleles <- myDataset$multilocus

## View the structure of amDataset object
unclass(myDataset)

## End(Not run)
```

---

amExampleData *Data sets to support the tutorials in the supplementary documentation and examples in the manual*

Description

amExample1 High quality data set
amExample2 Good quality data set
amExample3 Marginal quality data set
amExample4 Low quality data set
amExample5 Wildlife data set

Data sets 1 to 4 are simulated. Because the data are simulated, the individual from which the samples are derived is known. This is given in the column `knownIndividual`, and permits an assessment of how effective the software has been.

Data set 5 is a real dataset from a wildlife population that has been non-invasively sampled. The meta-data has been fabricated. It represents the output from allele calling software.

Usage

```
data(amExample1)
data(amExample2)
data(amExample3)
data(amExample4)
data(amExample5)
```

Format

Data frames with differing numbers of samples in rows, and alleles in columns. Missing data is represented as "-99".

Details

Note how in `amExample5` a single marker "Gender" has been combined with diploid loci.
Also note how in all data sets in diploid loci the lower length allele always comes first.
This pattern is typical in allele calling software.

---

amMatrix *Produce a dissimilarity matrix for pairs of multilocus genotypes*

Description

Given an amDataset object find the dissimilarities between pairs of multilocus genotypes, taking missing data into account.

Usage

```
amMatrix(
amDatasetFocal,
missingMethod = 2
)
```

Arguments

`amDatasetFocal`	An `amDataset` object. See `amDataset`.
`missingMethod`	The method used to determine the similarity of multilocus genotypes when data is missing. The default, (`missingMethod = 2`), is preferable in all cases. See `amMatrix`.

Details

This function is the behind-the-scenes workhorse of AlleleMatch, and typically will not be called by the user.

`missingMethod=2` is the recommended value, and the default, as it has performed better in simulations. In this method, missing data matches perfectly with missing data, while missing data matches partially with non-missing data.

`missingMethod = 1` is retained for experimental purposes. Here, missing data matches partially with missing and non-missing data.

Value

A distance/dissimilarity matrix of S3 class `amMatrix`.

Author(s)

Paul Galpern (<pgalpern@gmail.com>)

References

For a complete vignette, please access via the Data S1 Supplementary documentation and tutorials (PDF) located at <doi:10.1111/j.1755-0998.2012.03137.x>.

See Also

amPairwise, amUnique

Examples

```
## Not run:
data("amExample1")

## Produce amDataset object
myDataset <-
amDataset(
amExample1,
missingCode = "-99",
indexColumn = 1,
metaDataColumn = 2
)

## Produce dissimilarity matrix
dissimMatrix <- amMatrix(myDataset)

## End(Not run)
```

amPairwise *Pairwise matching of multilocus genotypes*

Description

Functions to perform a pairwise matching analysis of a multilocus genotype dataset, and review the output in formatted text or HTML. For each genotype in the focal dataset all genotypes in the comparison genotype are returned that match at or above a threshold matching score. The matching score is also known as the s-hat criterion (see the supplementary documentation). This is determined using amMatrix.

Usage

```
amPairwise(
amDatasetFocal,
amDatasetComparison = amDatasetFocal,
alleleMismatch = NULL,
matchThreshold = NULL,
missingMethod = 2
)

amHTML.amPairwise(
x,
htmlFile = NULL,
htmlCSS = amCSSForHTML()
)

amCSV.amPairwise(
x,
csvFile
)

  ## S3 method for class 'amPairwise'
summary(
    object,
    html = NULL,
    csv = NULL,
    ...
    )
```

Arguments

amDatasetFocal An amDataset object containing focal genotypes.

amDatasetComparison
Optional.
An amDataset object containing comparison genotypes.

	If not supplied, the focal dataset is also the comparison dataset (i.e., all focal dataset members are compared against one another).
`alleleMismatch`	Maximum number of mismatching alleles which will be tolerated when identifying individuals; also known as m-hat parameter. If specified, then `matchThreshold` should be omitted.
`matchThreshold`	Return comparison genotypes that match with the focal genotype at or above this score or similarity; also known as s-hat parameter.
`missingMethod`	Method used to determine the similarity of multilocus genotypes when data is missing. The default, (`missingMethod = 2`), is preferable in all cases. See `amMatrix`.
`object`, `x`	An `amPairwise` object.
`htmlFile`	HTML filepath to create. If `htmlFile = NULL`, a file is created in the operating system temporary directory and is then opened in the default browser.
`htmlCSS`	A string containing a valid cascading style sheet. A default style sheet is provided in `amCSSForHTML`. See `amCSSForHTML` for details of how to tweak this CSS.
`html`	If `html = NULL` or `html=FALSE`, formatted textual output is displayed on the console. If `html = TRUE`, the `summary.amPairwise` method produces and loads an HTML file in the default browser. `html` can also contain a path to a file where HTML output will be written.
`csvFile`, `csv`	CSV filepath to create containing giving a data frame representation of the pairwise matching results.
`...`	Additional arguments to `summary.amPairwise`.

Details

Pairwise matching of genotypes is a useful means to assess data quality and inspect for genotyping errors.

`matchThreshold` represents the similarity between two multilocus genotypes and can be thought of as a percentage similarity (or a Hamming's distance between two vectors) that has been corrected where missing data is present, such that missing data represents neither a match nor a mismatch but a "partial" match. See `amMatrix` for more discussion of this metric.

Value

`amPairwise` object or side effects: analysis summary written to an HTML file or to the console, or written to a CSV file.

Note

As `matchThreshold` is lowered, the size of the output increases rapidly. Typically analyses will not be very useful or manageable with thresholds below 0.7.

There is an additional side effect of `html = TRUE` (or of `htmlFile = NULL`). If required, there is a clean up of the operating system temporary directory where AlleleMatch temporary HTML files are stored. Files that match the pattern am*.html and are older 24 hours are deleted from this temporary directory.

Author(s)

Paul Galpern (<pgalpern@gmail.com>)

References

For a complete vignette, please access via the Data S1 Supplementary documentation and tutorials (PDF) located at <doi:10.1111/j.1755-0998.2012.03137.x>.

See Also

amDataset, amMatrix, amUnique

Examples

```
## Not run:
data("amExample5")

## Produce amDataset object
myDataset <-
amDataset(
amExample5,
missingCode = "-99",
indexColumn = 1,
metaDataColumn = 2,
ignoreColumn = "gender"
)

## Typical usage
myPairwise <-
amPairwise(
myDataset,
alleleMismatch = 2
)

## Display analysis as HTML in default browser
summary.amPairwise(
myPairwise,
html = TRUE
)

## Save analysis to HTML file
summary.amPairwise(
myPairwise,
html = "myPairwise.htm"
)
```

```
## Save analysis to CSV file
summary.amPairwise(
myPairwise,
csv = "myPairwise.csv"
)

## Display analysis as formatted text on the console
summary.amPairwise(myPairwise)

## Compare one dataset against a second
## Both must have same number of allele columns
## Here we create two datasets artificially from one for illustration purposes
myDatasetA <-
amDataset(
amExample5[sample(nrow(amExample5))[1:25], ],
missingCode = "-99",
indexColumn = 1,
ignoreColumn = 2
)
myDatasetB <-
amDataset(
amExample5[sample(nrow(amExample5))[1:100], ],
missingCode = "-99",
indexColumn = 1,
ignoreColumn = 2
)
myPairwise2 <-
amPairwise(
myDatasetA,
myDatasetB,
alleleMismatch = 3
)
summary.amPairwise(
myPairwise2,
html = TRUE
)

## End(Not run)
```

---

amPreCheck *Pre-Screen an amDataset Object for Severe Missing Data Loads*

Description

Analyzes missing data footprints and pairwise locus overlaps within an amDataset object to anticipate mathematical failures during subsequent clustering.

Usage

```
amPreCheck(amDatasetFocal)
```

Arguments

amDatasetFocal An object of class amDataset.

Value

Prints a quality control metric dashboard to the console. Also returns an invisible list containing summary percentages.

Author(s)

Todd Cross

Examples

```
  ## Not run:
    data(amExample1)
    techreps <- amDataset(amExample1, missingCode = "-99")
    amPreCheck(techreps)

## End(Not run)
```

---

amUnique *Identification of unique genotypes*

Description

Identifies unique genotypes and generates analysis output in formatted text, HTML, or CSV. Samples are clustered and matched based on their dissimilarity score (see amMatrix). Also calculated is the match probability, Psib, which is the probability that a sample is a sibling of a unique genotype (and therefore not a replicate sample) given the allele frequencies in a population consisting of only the unique genotypes (Wilberg & Dreher, 2004).

Usage

```
amUnique(
amDatasetFocal,
multilocusMap = NULL,
alleleMismatch = NULL,
matchThreshold = NULL,
cutHeight = NULL,
doPsib = "missing",
consensusMethod = 1,
verbose = TRUE
)

amHTML.amUnique(
x,
```

```
htmlFile = NULL,
htmlCSS = amCSSForHTML()
)

amCSV.amUnique(
x,
csvFile,
uniqueOnly = FALSE
)


  ## S3 method for class 'amUnique'
summary(
    object,
    html = NULL,
    csv = NULL,
    ...
    )
```

Arguments

amDatasetFocal	An amDataset object containing genotypes in which an unknown number of individuals are sampled multiple times
multilocusMap	Optional. A vector of integers or strings giving the mappings onto loci for all genotype columns in amDatasetFocal. When omitted, columns are assumed to be paired (i.e., diploid loci with alleles in adjacent columns). See details.
alleleMismatch	Optional. Maximum number of mismatching alleles which will be tolerated when identifying individuals; also known as m-hat parameter. If specified, then matchThreshold and cutHeight should be omitted. All three parameters are related. See details.
matchThreshold	Optional. Minimum dissimilarity score which constitutes a match when identifying individuals; also known as s-hat parameter. If specified, then alleleMismatch and cutHeight should be omitted; all three parameters are related. See details.
cutHeight	Optional. The cutHeight parameter used in dynamic tree cutting by amCluster; also known as d-hat parameter. If specified, then alleleMismatch and matchThreshold should be omitted. All three parameters are related. See details.

doPsib	String specifying how match probability should be calculated. See details.
consensusMethod	The method (an integer) used to determine the consensus multilocus genotype from a cluster of multilocus genotypes. See amCluster for details. Typically the default is adequate.
verbose	If verbose = TRUE, report the progress of the analysis to the console. Useful with datasets consisting of thousands of samples where progress may be slow.
object, x	An amUnique object.
htmlFile	HTML filepath to create. If htmlFile = NULL, a file is created in the operating system temporary directory and is then opened in the default browser.
htmlCSS	A string containing a valid cascading style sheet. A default style sheet is provided in amCSSForHTML. See amCSSForHTML for details of how to tweak this CSS.
html	If html = TRUE, the summary.amUnique method produces and loads an HTML file in the default browser. html can also contain a path to a file where HTML output will be written. Note that summary.amUnique does not produce formatted output for the console.
csvFile, csv	CSV filepath to create containing a representation of the amUnique analysis.
uniqueOnly	If uniqueOnly = TRUE, only the unique genotypes will be saved to a CSV, with no additional information associated with the analysis.
...	Additional arguments to summary.amUnique

Details

Only one of alleleMismatch, cutHeight, matchThreshold can be specified, as the three parameters are related.

alleleMismatch is the most intuitive way to understand how the identification of unique genotypes proceeds. For example, a setting of alleleMismatch = 4 implies that up to four alleles may be different for multiple samples to be representatives of the same individual. In practice, however, this value is only an approximation of the amount of mismatch that may be tolerated. This is because the clustering process used to identify unique genotypes, and the subsequent matching which identifies samples that match these unique genotypes is based on a dissimilarity metric or score (see amMatrix) that incorporates both allele mismatches and missing data. alleleMismatch is not used in analyses and is converted to this dissimilarity metric in the following manner: cutHeight which is parameter for amCluster and called from this function is cutHeight = alleleMismatch/(number of allele columns) and matchThreshold which is a parameter for amPairwise and also called from this function is matchThreshold = 1 - cutHeight.

Selecting the appropriate value for alleleMismatch, cutHeight, or matchThreshold is an important task. Use amUniqueProfile to assist in this process. Seethe Data S1 Supplementary documentation and tutorials (PDF) located at <doi:10.1111/j.1755-0998.2012.03137.x>

doPsib = "missing" is the default and specifies that match probability Psib should be calculated for samples that match unique genotypes and have no allele mismatches, but may differ by having missing data. doPsib = "all" specifies that Psib should be calculated for all samples that match unique genotypes. In this case, if allele mismatches occur, alleles are assumed to be missing at the mismatching loci.

multilocusMap is often not required, as amDataset objects will typically consist of paired columns of genotypes, where each pair is a separate locus. In cases where this is not the case (e.g., gender is in only one column), a map vector must be specified.

Example: amDataset consists of gender followed by 4 diploid loci in paired columns
multilocusMap = c(1, 2, 2, 3, 3, 4, 4, 5, 5)
or equally
multilocusMap=c("GENDER", "LOC1", "LOC1", "LOC2", "LOC2", "LOC3", "LOC4","LOC4")

For more information on selecting consensusMethod see amCluster. The default consensusMethod = 1 is typically adequate.

Value

amUnique object or side effects: analysis summary written to an HTML file or to the console, or written to a CSV file.

Note

There is an additional side effect of html = TRUE (or of htmlFile = NULL). If required, there is a clean up of the operating system temporary directory where AlleleMatch temporary HTML files are stored. Files that match the pattern am*.html and are older 24 hours are deleted from this temporary directory.

Author(s)

Paul Galpern (<pgalpern@gmail.com>)

References

For a complete vignette, please access via the Data S1 Supplementary documentation and tutorials (PDF) located at <doi:10.1111/j.1755-0998.2012.03137.x>.

Wilberg MJ, Dreher BP (2004) GENECAP: a program for analysis of multilocus genotype data for non-invasive sampling and capture-recapture population estimation. *Molecular Ecology Notes*, 4, 783-785.

See Also

amDataset, amMatrix, amPairwise, amCluster, amUniqueProfile

Examples

```
## Not run:
data("amExample2")

## Produce amDataset object
myDataset <-
amDataset(
amExample2,
missingCode = "-99",
indexColumn = 1,
ignoreColumn = 2
)

## Usage
## Optimal alleleMismatch parameter previously found using amUniqueProfile()
myUnique <-
amUnique(
myDataset,
alleleMismatch = 3
)

## Display analysis as HTML in default browser
summary.amUnique(
myUnique,
html = TRUE
)

## Save analysis to HTML file
summary.amUnique(
myUnique,
html = "myUnique.htm"
)

## Save analysis to a CSV file
summary.amUnique(
myUnique,
csv = "myUnique.csv"
)

## Save unique genotypes only to a CSV file
summary.amUnique(
myUnique,
csv = "myUnique.csv",
uniqueOnly = TRUE
)

## Data set with gender information
data("amExample5")

## Produce amDataset object
myDataset2 <-
amDataset(
```

```
amExample5,
missingCode = "-99",
indexColumn = 1,
metaDataColumn = 2
)

## Usage
## Optimal alleleMismatch parameter previously found using amUniqueProfile()
myUniqueProfile <-
amUnique(
myDataset2,
multilocusMap = c(1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 10,
11, 11),
alleleMismatch = 3
)


## End(Not run)
```

`amUniqueProfile` *Determine optimal parameter values for the identification of unique genotypes*

Description

Function to automatically run `amUnique` at a sequence of parameter values to determine an optimal setting, and optionally plot the result

Usage

```
amUniqueProfile(
amDatasetFocal,
multilocusMap = NULL,
alleleMismatch = NULL,
matchThreshold = NULL,
cutHeight = NULL,
guessOptimum = TRUE,
doPlot = TRUE,
consensusMethod = 1,
verbose = TRUE
)
```

Arguments

`amDatasetFocal`	An `amDataset` object containing genotypes in which an unknown number of individuals are sampled multiple times.

`multilocusMap` Optionally a vector of integers or strings giving the mappings onto loci for all genotype columns in amDatasetFocal.
When omitted, columns are assumed to be paired (i.e., diploid loci with alleles in adjacent columns).
See details.

`alleleMismatch` A vector giving a sequence, where elements give the maximum number of mismatching alleles which will be tolerated when identifying individuals; also known as the m-hat parameter.
If specified, then `matchThreshold` and `cutHeight` should be omitted; all three parameters are related.
See `amUnique` for details.

`matchThreshold` A vector giving a sequence, where elements give the minimum dissimilarity score which constitutes a match when identifying individuals; also known as the s-hat parameter.
If specified, then `alleleMismatch` and `cutHeight` should be omitted; all three parameters are related.
See `amUnique` for details.

`cutHeight` A vector giving a sequence, where elements give the `cutHeight` parameter used in dynamic tree cutting by `amCluster`; also known as the d-hat parameter.
If specified, then `alleleMismatch` and `matchThreshold` should be omitted; all three parameters are related.
See details.

`doPlot` If `doPlot = TRUE`, then a plot showing summary data from multiple runs of `amUnique` is produced.

`guessOptimum` If `guessOptimum = TRUE`, then will estimate the optimal value of the parameter being profiled by examining the profile for the first minimum associated with a drop in multiple matches as sensitivity to differences among samples decreases.

`consensusMethod`
The method (an integer) used to determine the consensus multilocus genotype from a cluster of multilocus genotypes.
See `amCluster` for details.

`verbose` If `verbose = TRUE`, then the progress of the profiling will be reported to the console.

Details

Selecting the appropriate value for `alleleMismatch`, `cutHeight`, or `matchThreshold` is an important task. Use this function to assist in this process. Typically the optimal value of any of these parameters is found where the number of multiple matches is minimized (the majority of samples are similar to only one unique genotype). Usually there is a minimum when these parameters are set to be very sensitive to differences among samples (i.e., `alleleMismatch` or `cutHeight` are 0, `matchThreshold` is 1). Simulations suggest that the next most sensitive minimum in multiple matches is the optimal value. This minimum will often be associated with a drop in multiple matches as sensitivity drops. For more discussion of this important step, see the Data S1 Supplementary documentation and tutorials (PDF) located at <doi:10.1111/j.1755-0998.2012.03137.x>.

Using `guessOptimum = TRUE` will attempt to estimate the location of this minimum and add it to the profile plot. Manual assessment of this estimate using the plot is strongly recommended.

If none of `alleleMismatch`, `cutHeight`, or `matchThreshold` is given, the function runs a sequence of values for `alleleMismatch` as follows: `seq(from = 0, to = floor(ncol(amDatasetFocal$multilocus) * 0.4), by = 1)`

`multilocusMap` is often not required, as `amDataset` objects will typically consist of paired columns of genotypes, where each pair is a separate locus. In cases where this is not the case (e.g., gender is given in only one column), a map vector must be specified.

Example: `amDataset` consists of gender followed by 4 diploid loci in paired columns
`multilocusMap = c(1, 2, 2, 3, 3, 4, 4, 5, 5)`
or equally
`multilocusMap=c("GENDER", "LOC1", "LOC1", "LOC2", "LOC2", "LOC3", "LOC4","LOC4")`

For more information on selecting `consensusMethod` see `amCluster`. The default `consensusMethod = 1` is typically adequate.

Value

A `data.frame` containing summary data from multiple runs of `amUnique`

Author(s)

Paul Galpern (<pgalpern@gmail.com>)

References

For a complete vignette, please access via the Data S1 Supplementary documentation and tutorials (PDF) located at <doi:10.1111/j.1755-0998.2012.03137.x>.

See Also

`amUnique`

Examples

```
## Not run:
data("amExample2")

## Produce amDataset object
myDataset <-
amDataset(
amExample2,
missingCode = "-99",
indexColumn = 1,
metaDataColumn = 2
)
```

```
## Usage (uncomment)
myUniqueProfile <-
amUniqueProfile(myDataset)

## Data set with gender information
data("amExample5")

## Produce amDataset object
myDataset2 <-
amDataset(
amExample5,
missingCode = "-99",
indexColumn = 1,
metaDataColumn = 2
)

## Usage
myUniqueProfile <-
amUniqueProfile(
myDataset2,
multilocusMap = c(1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 10,
11, 11))

## End(Not run)
```

Index

allelematch *(allelematch-package)*, 2
allelematch-package, 2
allelematch-tutorial *(allelematch-package)*, 2
amAlleleFreq, *3*, 3
amCluster, *3*, 5, *22, 23*, *26, 27*
amCSSForHTML, *7*, 10, *17*, *22*
amCSV.amCluster *(amCluster)*, 5
amCSV.amPairwise *(amPairwise)*, 16
amCSV.amUnique *(amUnique)*, 20
amDataset, *2*, *8*, 11, *14*, *18*, *23*
amExample1 *(amExampleData)*, 13
amExample2 *(amExampleData)*, 13
amExample3 *(amExampleData)*, 13
amExample4 *(amExampleData)*, 13
amExample5 *(amExampleData)*, 13
amExampleData, *12, 13*, 13
amHTML.amCluster *(amCluster)*, 5
amHTML.amPairwise, *10*
amHTML.amPairwise *(amPairwise)*, 16
amHTML.amUnique, *10*
amHTML.amUnique *(amUnique)*, 20
amMatrix, *3*, *6–8*, *14*, 14, *16–18*, *20*, *22, 23*
amPairwise, *3*, *8*, *12, 13*, *15*, 16, *22, 23*
amPreCheck, *2*, 19
amUnique, *3–5*, *8*, *13*, *15*, *18*, 20, *26, 27*
amUniqueProfile, *3*, *8*, *23*, 25

cutreeHybrid, *6*

hclust, *6*

print.amAlleleFreq *(amAlleleFreq)*, 3
print.amDataset *(amDataset)*, 11

summary.amCluster *(amCluster)*, 5
summary.amPairwise *(amPairwise)*, 16
summary.amUnique *(amUnique)*, 20