# Package 'mlPhaser'

February 20, 2015

**Type** Package

**Title** Multi-Locus Haplotype Phasing

**Version** 0.01

**Date** 2012-09-03

**Author** Dave T. Gerrard

**Maintainer** Dave T. Gerrard <david.gerrard@manchester.ac.uk>

**Description** Phase haplotypes from genotypes based on a list of known haplotypes. Suited to highly diverse loci such as HLA.

**License** GPL (>= 2)

**LazyLoad** yes

**URL** https://github.com/davetgerrard/mlPhaser

**Repository** CRAN

**Date/Publication** 2012-09-04 13:38:46

**NeedsCompilation** no

## R topics documented:

---

`mlPhaser-package` *mlPhaser*

---

## Description

| | |
|---|---|
| Package: | mlPhaser |
| Type: | Package |
| Version: | 0.01 |
| Date: | 2012-08-28 |
| Author: | Dave T. Gerrard <david.gerrard@manchester.ac.uk> |
| License: | GPL (>= 2) |
| LazyLoad: | yes |

## Details

mlPhaser

Some text

The main functions are: `phaseReport`, `getValidHaploGroups`,

...

---

`getHaploGroupProb` *Get haplotype group probability*

---

## Description

Combine probabilities across a haplotype group

## Usage

```
getHaploGroupProb(haploGroup, haploFreqs,
  method = "basic", returnLog = FALSE)
```

## Arguments

| | |
|---|---|
| `haploGroup` | A list of haplotypes. Only the names attribute is important. |
| `haploFreqs` | The frequencies of haplotypes as a named vector. |
| `method` | Only one method currently ("basic"). |
| `returnLog` | Whether to output the combined probability as a natural log. Default=FALSE. |

### Details

Each haplotype might have a probability of being found. e.g. the population frequency. This function combines probabilities of a group of haplotypes

### Value

A numeric likelihood representing the combined probability across a group.

---

## getValidHaploGroups *Get haplo groups for a genotype*

---

### Description

Get all valid groups of haplotypes that fully explain a genotype.

### Usage

```
getValidHaploGroups(genotype, haplotypes)
```

### Arguments

| | |
|---|---|
| genotype | The genotype in question. Can be data.frame or list of lists format |
| haplotypes | The set of candidate haplotypes. |

### Details

Wrapper function to set up and control the recursive search for groups of haplotyps, each of which are consistent with the genotype in question. Makes use of recursion via the function recurseHaplos

### Value

A list of valid haplotype groups (each itself a list of haplotypes).

### See Also

phaseReport

### Examples

```
# create a data frame to store alleles of haplotypes. Columns are loci.
haplotypes <- data.frame( A= c("a","b","c","a","b","c","b"),
B= c("a","b","c","b","c","a","a"),
C= c("a","b","c","b","c","a","a") )
# give the haplotypes sensible names as rownames.
rownames(haplotypes) <- apply(haplotypes, 1, paste,sep="" , collapse="")
# load a genotype as a table
thisGenotype <- data.frame(A.1="a", A.2="b", B.1="a", B.2="b",C.1="a", C.2="b")
```

```
# find groups of haplotypes as a list of lists
my.valid.groups <- getValidHaploGroups(thisGenotype,haplotypes)
# look at the list structure of the valid groups list
str(my.valid.groups)
# see phaseReport() for more friendly function
```

---

`listGenoToTable` *Genotype list to genotype table*

---

## Description

Converts a list of genotypes to a table with several columns per locus.

## Usage

```
listGenoToTable(genoList)
```

## Arguments

| | |
|---|---|
| `genoList` | A list of lists. One entry per genotype (sample) each containing a list of loci to store the alleles. |

## Details

The multiple columns per locus are differentiated with a numerical suffic. e.g. locA.1, locA.2

## Value

A data.frame a row for each genotype and n colums for each locus (where n is ploidy of locus)

---

`listHaploToTable` *Haplotype list to Haplotype table.*

---

## Description

Converts a set of haplotypes from list format to table format

## Usage

```
listHaploToTable(haploList)
```

## Arguments

| | |
|---|---|
| `haploList` | A list of lists. One top level element per haplotype. Each haplotype should be named and have the same set of loci as a sublist. |

### Details

Each list element becomes a row. Each locus becomes a column.

### Value

A set of haplotypes in table format as a data.frame

---

## listValidHaplotypes *List valid haplotypes*

---

### Description

Select valid haplotypes (from a list) for a genotype

### Usage

```
listValidHaplotypes(genotype, haplotypes, ploidy = 2)
```

### Arguments

| | |
|---|---|
| genotype | The genotype. Can be in data.frame or list of list format. |
| haplotypes | The haplotypes. Can be in data.frame or list of lists format. |
| ploidy | How many alleles per locus. Default=2. |

### Details

Tests a set of haplotypes against a genotype to see if they are contained within.

### Value

A list of valid haplotypes (each as a list).

---

## phaseReport *Best/all haplotype groups for a genotype*

---

### Description

Attempts to find best/all haplotype groups that fully explain observed multi-locus genotypes.

### Usage

```
phaseReport(genotypes, haplotypes, haploFreqs,
  outFormat = "all")
```

## Arguments

| | |
|---|---|
| genotypes | The table/list of genotypes |
| haplotypes | The table/list of candidate haplotypes |
| haploFreqs | The frequencies of haplotypes as a named vector. |
| outFormat | Whether to output all valid haplotype groups or just the best (based on joint probability). |

## Details

This wrapper function takes a set of genotypes, a set of haplotypes and a set of haplotype frequencies and attempts to report either all groups or just the single most likely group of known haplotypes that fully explains each observed genotype.

## Value

A data.frame of results...

## See Also

getValidHaploGroups

## Examples

```
# create a data frame to store alleles of haplotypes. Columns are loci.
haplotypes <- data.frame( A= c("a","b","c","a","b","c","b"),
B= c("a","b","c","b","c","a","a"),
C= c("a","b","c","b","c","a","a") )
# give the haplotypes sensible names as rownames.
rownames(haplotypes) <- apply(haplotypes, 1, paste,sep="" , collapse="")
# Create a named vector of haplotype frequencies.
haploFreqs <- c(0.4, 0.3, 0.15, 0.07,0.05, 0.02, 0.01)
names(haploFreqs) <- rownames(haplotypes)
# load a genotype as a table
thisGenotype <- data.frame(A.1="a", A.2="b", B.1="a", B.2="b",C.1="a", C.2="b")
phaseReport(thisGenotype,haplotypes)
# use haplotype frequencies to rank candidate haplotype groups.
phaseReport(thisGenotype,haplotypes, haploFreqs)
# return only the best haplotype group for each genotype.
phaseReport(thisGenotype,haplotypes, haploFreqs, outFormat="top")

# simulate a set of genootypes
my.genotypes <- simGenoFromHaplo(haploTable=haplotypes, haploFreqs=haploFreqs , n=20, ploidy=2)
# get phase report on all genotypes
phaseReport(my.genotypes,haplotypes, haploFreqs, outFormat="all") # outFormat="all" is the default
phaseReport(my.genotypes,haplotypes, haploFreqs, outFormat="top")
```

## printHaploGroups *Print haplotype groups*

### Description

Prints haplotype groups from a list.

### Usage

```
printHaploGroups(haploListOfLists)
```

### Arguments

| | |
|---|---|
| haploListOfLists | A list of haplotype groups. |

### Details

Works through a list of haplotype groups (quite a complex list structure) and prints out one group per line. For presentation only, results cannot be re-used.

### Value

Nothing. Just prints.

## printHaploProbs *Print haplo group probabilities*

### Description

Print out haplotype groups and their relative probabilities

### Usage

```
printHaploProbs(namedHaploGroups, haploFrequencies)
```

### Arguments

| | |
|---|---|
| namedHaploGroups | A list of haplotype groups (each a list) that explain a genotype. |
| haploFrequencies | A named numeric vector giving the probabilities of haplotypes. Names store the haplotype names. |

## Details

This was a first attempt to order competing haplotype groups. It only prints and does not return a useable object.

## Value

Nothing. Prints only

---

`recurseHaplos` *Recurse to get haplo groups*

---

## Description

Recursive function to extract valid groups of haplotypes explaining a genotype

## Usage

```
recurseHaplos(validHaplotypes, remGenotype, group)
```

## Arguments

| | |
|---|---|
| `validHaplotypes` | A list of haplotypes to choose from |
| `remGenotype` | The remaining part of the genotype |
| `group` | The valid group to this point. |

## Details

This recursive function subtracts haplotypes from a genotypes to find 'groups' of haplotypes that can fully explain a genotype. To make the function general and cope with ploidy > 2, I made it recursive. It will keep going until it has run out of genotype and/or it has run out of valid haplotypes. This should probably stay as an internal function because of its recursive nature. N.B. requires access to a globally accessible storage variable: validHaploGroups

## Value

NULL. N.B. requires access to a globally accessible storage variable: validHaploGroups

## reduceRedundantList *Remove redundant haplotype groups*

### Description

Removes redundant groups of haplotypes from a common list.

### Usage

```
reduceRedundantList(startList)
```

### Arguments

| | |
|---|---|
| startList | A list of haplotype groups (each is a list of haplotypes). |

### Details

The recursive method recurseHaplos of finding groups of consistent haplotypes does not differentiate, re-arranged versions of the same set. e.g. keeps aaa/bbb AND bbb/aaa. This function removes that redundancy from the results. uses the length of intersect to determine if two lists contain all the same elements.

### Value

A list of haplotype groups but each group is unique.

## remGeno *Extract haplotype from genotype*

### Description

Attempts to extract a haplotype from a genotype

### Usage

```
remGeno(haplotype, genotypeList)
```

### Arguments

| | |
|---|---|
| haplotype | The haplotype to be removed |
| genotypeList | The genotype in list of lists format. |

### Details

Tries to extract a single haplotype from a compound genotype and return, amongst other things, the remainder genotype.

## Value

A list giving the original haplotype extracted (haplotype), a table of TRUE/FALSE for each locus with TRUE if the allele was successfully extracted (passTable), and a list giving the genotype remaining after extraction (remList).

---

`simGenoFromHaplo` *Simulate genotypes*

---

## Description

Simulates genotypes from a table of haplotypes.

## Usage

```
simGenoFromHaplo(haploTable, haploFreqs, n = 1,
  ploidy = 2)
```

## Arguments

| | |
|---|---|
| `haploTable` | The list of haplotypes in table format |
| `haploFreqs` | A named vector of haplotype frequencies. |
| `n` | How many genotypes to simulate. |
| `ploidy` | How many alleles per locus. Default = 2. |

## Details

Simulates n genotypes from a table of haplotypes. Genotypes will include one allele per ploidy level.

## Value

A data.frame of genotypes. Each locus will have multiple colums as per the ploidy level.

## Examples

```
# create a data frame to store alleles of haplotypes. Columns are loci.
haplotypes <- data.frame( A= c("a","b","c","a","b","c","b"),
B= c("a","b","c","b","c","a","a"),
C= c("a","b","c","b","c","a","a") )
# give the haplotypes sensible names as rownames.
rownames(haplotypes) <- apply(haplotypes, 1, paste,sep="" , collapse="")
# Create a named vector of haplotype frequencies.
haploFreqs <- c(0.4, 0.3, 0.15, 0.07,0.05, 0.02, 0.01)
names(haploFreqs) <- rownames(haplotypes)

# simulate a set of genotypes
my.genotypes <- simGenoFromHaplo(haploTable=haplotypes, haploFreqs=haploFreqs , n=20, ploidy=2)
```

## tableGenoToList *Convert genotype table to list of lists*

### Description

Converts a table of genotypes to a list of lists, one sub-list per genotype.

### Usage

```
tableGenoToList(genoTable, locusNames)
```

### Arguments

| | |
|---|---|
| genoTable | A data.frame containg genotypes. One row per genotype. Multiple columns per locus as per the ploidy. |
| locusNames | A character vector of locus names. |

### Details

Converts a table of genotypes to a list of lists, one sub-list per genotype.

### Value

genotypes as a list of lists

## tableHaploToList *Haplotype table to haplotype list*

### Description

Converts a data.frame of haplotypes to a list of lists with haplotypes at the top level and list of loci (with their alleles) beneath.

### Usage

```
tableHaploToList(haploTable,
  locusNames = colnames(haploTable))
```

### Arguments

| | |
|---|---|
| haploTable | A data.frame of alleles making up the haplotypes. One columm per locus, one row per haplotype. The rownames should contain the haplotype ids. |
| locusNames | A character vector giving the names of the loci which should match the column names of the haploTable |

### Details

Summary paragraph outlining method

### Value

Haplotypes as a list of lists.

---

`testHaploInGeno` *Test genotype for presence of haplotype*

---

### Description

Test if a genotype contains a haplotype.

### Usage

```
testHaploInGeno(haplotype, genotypeList)
```

### Arguments

| | |
|---|---|
| `haplotype` | The haplotype as a one line data.frame |
| `genotypeList` | The genotype as a list of lists. |

### Details

An early implementation to test if a genotype contained a haplotype. N.B. I don't think this is used anymore.

### Value

TRUE/FALSE if haplotype is present in the genotype

# Index