Package 'pedbuildr'

March 16, 2021

Title Pedigree Reconstruction

Version 0.2.1

Description Reconstruct pedigrees from genotype data, by optimising the likelihood over all possible pedigrees subject to given restrictions. Tailor-made plots facilitate evaluation of the output. This package is part of the 'ped suite' ecosystem for pedigree analysis. In particular, it imports 'pedprobr' for calculating pedigree likelihoods and 'forrel' for estimating pairwise relatedness.

Encoding UTF-8

Language en-GB

LazyData true

License GPL-3

Depends R (>= 3.5.0), pedtools (>= 0.9.6)

Imports pedprobr, forrel (>= 1.3.0), glue

URL https://github.com/magnusdv/pedbuildr

BugReports https://github.com/magnusdv/pedbuildr/issues

RoxygenNote 7.1.1

Suggests testthat

NeedsCompilation no

Author Magnus Dehli Vigeland [aut, cre]
(<https://orcid.org/0000-0002-9134-4962>)

Maintainer Magnus Dehli Vigeland <m.d.vigeland@medisin.uio.no>

Repository CRAN

Date/Publication 2021-03-16 09:00:07 UTC

R topics documented:

`buildPeds` *Build a list of pedigrees*

Description

Build all pedigrees between a set of individuals, subject to given restrictions.

Usage

```
buildPeds(
  labs,
  sex,
  extra = "parents",
  age = NULL,
  knownPO = NULL,
  allKnown = FALSE,
  notPO = NULL,
  noChildren = NULL,
  connected = TRUE,
  linearInb = TRUE,
  maxLinearInb = NULL,
  sexSymmetry = TRUE,
  verbose = TRUE
)
```

Arguments

`labs`	A character vector of ID labels.
`sex`	A vector of the same length as `labs`, with entries 1 (male) or 2 (female).
`extra`	Either the word "parents", or a nonnegative integer. See details.
`age`	A numeric or character vector. If numeric, and `age[i] < age[j]`, then individual `i` will not be an ancestor of individual `j`. The numbers themselves are irrelevant, only the partial ordering. Note that no interpretation is made about individuals of equal age. Alternatively `age` may be a character vector of inequalities, e.g., `age = c("1>2","1>3")`. This syntax allows finer control than the numeric version.
`knownPO`	A list of vectors of length 2, containing the ID labels of pairs known to be parent-offspring. By default, both directions are considered; use `age` to force a specific direction.
`allKnown`	A logical. If TRUE, no other pairs than `knownPO` will be assigned as parent-offspring. If FALSE (default), all pairs except those in `notPO` are treated as potential parent-offspring.
`notPO`	A list of vectors of length 2, containing the ID labels of pairs known *not* to be parent-offspring.

noChildren	A vector of ID labels, indicating individuals without children of their own.
connected	A logical. If TRUE (default), only connected pedigrees are returned.
linearInb	Either TRUE (allow any linear inbreeding), FALSE (disallow linear inbreeding) or a nonnegative integer indicating the maximum separation linearly related spouses. For example, linearInb = 1 allows mating between parent and child, but not between grandparent and grandchild (or more distant).
maxLinearInb	Deprecated; replaced by linearInb.
sexSymmetry	A logical. If TRUE (default), pedigrees which are equal except for the gender distribution of the *added* parents, are regarded as equivalent, and only one of each equivalence class is returned. Example: paternal vs. maternal half sibs.
verbose	A logical.

Details

The parameter extra controls which of two algorithms are used to create the pedigree list.

If extra is a nonnegative integer, it determines the number of extra individuals allowed in the iterative pedigree construction. These extras start off with undetermined sex, meaning that both males and females are used. It should be noted that the final pedigrees may contain additional extras, since missing parents are added at the end.

If extra is the word "parents", the algorithm is not iterative. It first generates all directed acyclic graphs between the original individuals. Then their parents are added and merged in all possible ways. This option has the advantage of not requiring an explicit/ad hoc number of "extras", but works best in smaller cases.

Value

A list of pedigrees. Each element is a ped object or a list of such.

Examples

```
# Showing off a few of the options
plist = buildPeds(1:3, sex = c(1,2,1), extra = 1, knownPO = list(1:2),
                  age = "1 > 2", linearInb = FALSE)
stopifnot(length(plist) == 12)


# Slightly different output with `extra = "parents"`
plist2 = buildPeds(1:3, sex = c(1,2,1), extra = "parents", knownPO = list(1:2),
                   age = "1 > 2", linearInb = FALSE)
stopifnot(length(plist2) == 8)
```

pedbuildr *pedbuildr: Pedigree reconstruction*

Description

Reconstruct pedigrees from genotype data, by optimising the likelihood over all possible pedigrees subject to given restrictions. Tailor-made plots facilitate evaluation of the output. This package is part of the 'ped suite' ecosystem for pedigree analysis in R. In particular, it imports 'pedprobr' for calculating pedigree likelihoods and 'forrel' for estimating pairwise relatedness.

reconstruct *Pedigree reconstruction*

Description

Reconstruct the most likely pedigree from genotype data.

Usage

```
reconstruct(
  x,
  ids,
  extra = "parents",
  alleleMatrix = NULL,
  loci = NULL,
  pedlist = NULL,
  inferPO = FALSE,
  sex = NULL,
  age = NULL,
  knownPO = NULL,
  allKnown = FALSE,
  notPO = NULL,
  noChildren = NULL,
  connected = TRUE,
  linearInb = TRUE,
  maxLinearInb = NULL,
  sexSymmetry = TRUE,
  sortResults = TRUE,
  founderInb = 0,
  verbose = TRUE
)
```

Arguments

x	A pedtools::ped object or a list of such.
ids	A vector of ID labels from x. By default, the genotyped members of x are used.
extra	Either the word "parents", or a nonnegative integer. See details.
alleleMatrix	A matrix with two columns for each marker.
loci	A list of marker attributes.
pedlist	A list of pedigrees. If NULL, buildPeds() is used to generate a list.
inferPO	A logical. If TRUE, an initial stage of pairwise IBD estimation is done, in order to infer certain parent-child pairs, as well as certain *non*-parent-child pairs. When this option is used, arguments to knownPO and notPO are ignored.
sex	A vector of the same length as labs, with entries 1 (male) or 2 (female).
age	A numeric or character vector. If numeric, and age[i] < age[j], then individual i will not be an ancestor of individual j. The numbers themselves are irrelevant, only the partial ordering. Note that no interpretation is made about individuals of equal age. Alternatively age may be a character vector of inequalities, e.g., age = c("1>2","1>3"). This syntax allows finer control than the numeric version.
knownPO	A list of vectors of length 2, containing the ID labels of pairs known to be parent-offspring. By default, both directions are considered; use age to force a specific direction.
allKnown	A logical. If TRUE, no other pairs than knownPO will be assigned as parent-offspring. If FALSE (default), all pairs except those in notPO are treated as potential parent-offspring.
notPO	A list of vectors of length 2, containing the ID labels of pairs known *not* to be parent-offspring.
noChildren	A vector of ID labels, indicating individuals without children of their own.
connected	A logical. If TRUE (default), only connected pedigrees are returned.
linearInb	Either TRUE (allow any linear inbreeding), FALSE (disallow linear inbreeding) or a nonnegative integer indicating the maximum separation linearly related spouses. For example, linearInb = 1 allows mating between parent and child, but not between grandparent and grandchild (or more distant).
maxLinearInb	Deprecated; replaced by linearInb.
sexSymmetry	A logical. If TRUE (default), pedigrees which are equal except for the gender distribution of the *added* parents, are regarded as equivalent, and only one of each equivalence class is returned. Example: paternal vs. maternal half sibs.
sortResults	A logical. If TRUE, the output is sorted so that the most likely pedigree comes first.
founderInb	A number in the interval [0,1], used as background inbreeding level in all founders.
verbose	A logical.

Details

The parameter `extra` controls which of two algorithms are used to create the pedigree list.

If `extra` is a nonnegative integer, it determines the number of extra individuals allowed in the iterative pedigree construction. These extras start off with undetermined sex, meaning that both males and females are used. It should be noted that the final pedigrees may contain additional extras, since missing parents are added at the end.

If `extra` is the word "parents", the algorithm is not iterative. It first generates all directed acyclic graphs between the original individuals. Then their parents are added and merged in all possible ways. This option has the advantage of not requiring an explicit/ad hoc number of "extras", but works best in smaller cases.

Value

An object of class `pedrec`, which is essentially list with the following entries:

- `pedlist`: A list of pedigrees, either built by `buildPeds()` or as supplied in the input argument `pedlist`. If `sortResults = TRUE`, the list is sorted so that the most likely pedigrees come first
- `logliks`: A numerical vector of pedigree log-likelihoods
- `kappa`: A data frame with pairwise estimates (if `inferPO = TRUE`)
- `alleleMatrix`: A matrix of marker alleles
- `loci`: A list of marker locus attributes
- `errPeds`: A list of pedigrees for which the likelihood calculation failed
- `errIdx`: The indices of pedigrees in `errPeds` as elements of `pedlist`

Examples

```
#-----------------
# Example 1: Trio
#-----------------

data(trioData)

x = as.ped(trioData, locusAttributes = "snp-12")
summary(x)

res = reconstruct(x, inferPO = TRUE, age = "1 > 2", linearInb = FALSE)

# Plot most likely pedigrees
plot(res, top = 6)


### Alternative workflow: Extract data manually...
als  = getAlleles(x)
loci = getLocusAttributes(x)
sex  = getSex(x)

# ...and then reconstruct
res2 = reconstruct(alleleMatrix = als, loci = loci, sex = sex,
```

```
                    inferPO = TRUE, age = "1 > 2", linearInb = FALSE)

stopifnot(identical(res, res2))

#--------------------
# Example 2: Siblings
#--------------------

ids = c("s1", "s2")
y = nuclearPed(children = ids)

# Simulate data
y = forrel::markerSim(y, N = 50, ids = ids, seed = 123)

# Reconstruct and plot
res3 = reconstruct(y, connected = FALSE)
res3
plot(res3)
```

`trioData` *Reconstruction example with three individuals*

Description

This dataset contains simulated genotypes for 3 males at 100 SNP markers.

Usage

```
trioData
```

Format

A data frame with 3 rows and 104 columns. The first 4 columns contain pedigree info:

- `id`: Individual ID
- `fid`: Paternal ID, where 0 means missing father
- `mid`: Maternal ID, where 0 means missing mother
- `sex`: 1 = male; 2 = female

The remaining 100 columns contain the genotypes, in the form a/b, where a and b are the observed alleles (labelled 1 and 2).

Index