Package: pedbuildr (via r-universe)

September 7, 2024

Title Pedigree Reconstruction

Version 0.3.0.9000

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 'pedsuite' ecosystem for pedigree analysis. In particular, it imports 'pedprobr' for calculating pedigree likelihoods and 'forrel' for estimating pairwise relatedness.

License GPL-3

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

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

Depends pedtools (>= 2.2.0), R (>= 4.1.0)

Imports forrel (>= 1.5.0), glue, pedmut, pedprobr, ribd

Suggests testthat

Encoding UTF-8

Language en-GB

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Repository https://magnusdv.r-universe.dev

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

RemoteRef HEAD

RemoteSha 953063b1560d6e30bef2e7e8a5e66d8c618976c6

Contents

`buildPeds` *Build a list of pedigrees*

Description

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

Usage

```
buildPeds(
  labs,
  sex = 1,
  extra = "parents",
  age = NULL,
  knownPO = NULL,
  knownSub = NULL,
  allKnown = FALSE,
  notPO = NULL,
  noChildren = NULL,
  connected = TRUE,
  maxInbreeding = 1/16,
  linearInb = FALSE,
  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" (default), or a non-negative 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. (No inference is made about individuals of equal age.) Alternatively, for finer control, `age` may be a character vector of inequalities, e.g., `age = c("1>2", "1>3")`.
`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.
`knownSub`	A `ped` object involving a subset of the `labs` individuals.
`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.
maxInbreeding	A single numeric indicating the highest permitted inbreeding coefficient. Default: 1/16 (as with first-cousin parents.)
linearInb	A parameter controlling the maximum separation of linearly related spouses. Either TRUE (allow all linear inbreeding), FALSE (disallow all) or a non-negative integer. For example, `linearInb = 1` allows parent/child mating, but not grandparent/grandchild or more distant linear relatives. Default: FALSE.
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 (possibly disconnected) pedigrees.

Examples

```
# Two individuals + 1 extra
plist = buildPeds(1:2, extra = 1, age = "1>2")
plot(plist)

# Allow disconnected
plist2 = buildPeds(1:2, extra = 1, age = "1>2", connected = FALSE)
plot(plist2, frames = TRUE)

# Note that full sibs require 2 extras
plist3 = buildPeds(1:2, extra = 2, age = "1>2")
plot(plist3)

# With 2 extras, allowing any inbreeding
plist4 = buildPeds(1:2, extra = 2, age = "1>2", maxInbreeding = 1)
plot(plist4)
```

```
# Full sibs are also included when `extra = "parents"`
plist5 = buildPeds(1:2, extra = "parents", age = "1>2")
plot(plist5)
```

`reconstruct` *Pedigree reconstruction*

Description

Reconstructs 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,
  knownSub = NULL,
  allKnown = FALSE,
  notPO = NULL,
  noChildren = NULL,
  connected = TRUE,
  maxInbreeding = 1/16,
  linearInb = FALSE,
  sexSymmetry = TRUE,
  sortResults = TRUE,
  founderInb = 0,
  numCores = 1,
  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" (default), or a non-negative integer. See Details.
`alleleMatrix`	A matrix with two columns for each marker. By default extracted from `x`

loci	A list of marker attributes. By default extracted from x.
pedlist	A list of pedigrees to optimise over. If NULL, buildPeds() is used to generate a list.
inferPO	A logical. If TRUE, an initial stage of pairwise IBD estimation is done to infer high-confidence parent/child pairs, and also *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. (No inference is made about individuals of equal age.) Alternatively, for finer control, age may be a character vector of inequalities, e.g., age = c("1>2", "1>3").
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.
knownSub	A ped object involving a subset of the labs individuals.
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.
maxInbreeding	A single numeric indicating the highest permitted inbreeding coefficient. Default: 1/16 (as with first-cousin parents.)
linearInb	A parameter controlling the maximum separation of linearly related spouses. Either TRUE (allow all linear inbreeding), FALSE (disallow all) or a non-negative integer. For example, linearInb = 1 allows parent/child mating, but not grandparent/grandchild or more distant linear relatives. Default: FALSE.
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 (default), 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. Default: 0.
numCores	A positive integer. The number of cores used in parallelisation. Default: 1.
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:

- `labs`: The individual labels as given in `ids`.
- `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
#-----------------

# Built-in dataset `trioData`
x = singletons(1:3) |>
  setMarkers(alleleMatrix = trioData, locusAttributes ="snp12")

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

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


#--------------------
# Example 2: Siblings
#--------------------
library(forrel)

ids = c("s1", "s2")

# Create pedigree and simulate profiles with 20 STR markers
y = nuclearPed(children = ids) |>
  profileSim(markers = NorwegianFrequencies[1:20], ids = ids, seed = 123)

# Reconstruct allowing 2 extra individuals and any inbreeding
```

```
res2 = reconstruct(y, extra = 2, maxInb = 1)
plot(res2)

# With mutation modelling
y = setMutmod(y, model = "equal", rate = 0.01)
res3 = reconstruct(y, extra = 2, maxInb = 1)
plot(res3)
```

`trioData` *Reconstruction example with three individuals*

Description

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

Usage

```
trioData
```

Format

A matrix with 3 rows and 100 columns. Each entry contains a genotype in the form a/b.

Examples

```
trioData[, 1:10]

x = singletons(1:3) |>
  setMarkers(alleleMatrix = trioData, locusAttributes ="snp12")

x
```

`Tutankhamun` *Pedigree of Tutankhamun*

Description

A reconstructed pedigree of the Egyptian Pharaoh Tutankhamun, with genotypes for 8 STR markers, as published by Hawass et al.

Usage

```
Tutankhamun
```

Format

A data frame with 7 rows and 12 columns:

- `id`,`fid`,`mid`,`sex`: Pedigree columns in standard format
- `D13S317`, ...: Genotype columns for 8 markers

Source

Hawass et al. *Ancestry and pathology in King Tutankhamun's family*. Jama (2010).

Examples

```
# Pedigree as published
plot(Tutankhamun)

# Simple reconstruction, assuming all directly related
res = reconstruct(Tutankhamun, extra = 0, inferPO = TRUE, maxInbreeding = 1)
plot(res, top = 4)

# Published ped is most likely (with these assumptions)
identical(res[[1]], Tutankhamun)
```

Index