Package ‘Familias’

January 20, 2025

Type Package

Title Probabilities for Pedigrees Given DNA Data

Version 2.6.2

Description An interface to the core 'Familias' functions which are programmed in C++. The implementation is described in Egeland, Mostad and Olaisen (1997) <doi:10.1016/S1355-0306(97)72202-0> and Simonsson and Mostad (2016) <doi:10.1016/j.fsigen.2016.04.005>.

License GPL-3

URL `https://www.familias.name/openfamilias.html`

Depends kinship2, R (>= 3.4.0), Rsolnp

Encoding UTF-8

LazyLoad yes

NeedsCompilation yes

Author Petter Mostad [aut],
Thore Egeland [aut, cre],
Ivar Simonsson [aut]

Maintainer Thore Egeland <`Thore.Egeland@nmbu.no`>

Repository CRAN

Date/Publication 2025-01-08 09:30:02 UTC

Contents

Familias-package — *Familias: Inferring paternity and identification based on DNA data*

Description

This package provides an R interface to the Windows program Familias (https://familias.no), for calculating probabilities in forensic family genetics. The core source code (in C++) of the Windows program is contained in the package.

Details

The main purpose of this package is to provide an R interface to the calculation of likelihood ratios (LRs) in Familias. For other forensic pedigree analyses and visualisations we recommend the 'pedsuite' packages: https://magnusdv.github.io/pedsuite/. In particular the package 'pedFamilias' facilitates conversion of .fam files into the 'pedsuite' format.

Author(s)

Petter Mostad <mostad@chalmers.se> and Thore Egeland <Thore.Egeland@nmbu.no>.

References

For more information, see https://familias.name/.

Examples

```
persons <- c("mother", "daughter", "AF")

# Pedigrees
ped1 <- FamiliasPedigree(id = persons,
                         dadid = c(NA, "AF", NA),
                         momid = c(NA, "mother", NA),
                         sex = c("female", "female", "male"))

ped2 <- FamiliasPedigree(id = c(persons, "TF"),
                         dadid = c(NA, "TF", NA, NA),
                         momid = c(NA, "mother", NA, NA),
                         sex = c("female", "female", "male", "male"))

ped3 <- FamiliasPedigree(id = c(persons, "TF", "gf", "gm"),
                         dadid = c(NA, "TF", "gf", "gf", NA, NA),
                         momid = c(NA, "mother", "gm", "gm", NA, NA),
                         sex = c("female", "female", "male", "male", "male", "female"))
op <- par(mfrow = c(1,3))
plot(ped1); title("ped1: AF is father")
plot(ped2); title("ped2: AF is unrelated")
plot(ped3); title("ped3: AF is uncle")
par(op)
```

```
mypedigrees <- list(isFather = ped1, unrelated=ped2, isUncle = ped3)

# Loci
locus1 <- FamiliasLocus(frequencies = c(0.1, 0.2, 0.3, 0.4),
                        allelenames = c("A", "B", "C", "D"),
                        name = "locus1")
locus2 <- FamiliasLocus(c(0.2, 0.3, 0.5),
                        c(17, 18, 19),
                        "loc2",
                        femaleMutationRate = 0.05)
myloci <- list(locus1, locus2)

# Genotype data
datamatrix <- data.frame(locus1.1 = c("A", "A", "A"),
                         locus1.2 = c("B", "B", "C"),
                         locus2.1 = c(17, 19, 19),
                         locus2.2 = c(18, 18, 18),
                         row.names = persons)

# Calculate LR
## Not run:
FamiliasPosterior(mypedigrees, myloci, datamatrix)

## End(Not run)
```

`FamiliasLocus` *Creates an object with information on a locus, including its mutation matrices*

Description

The user provides input needed to define an autosomal locus (also called system or forensic marker) to be used for pedigree calculations. The input is checked and if no errors are found a list with class `FamiliasLocus` is returned containing the information.

Usage

```
FamiliasLocus(frequencies,
              allelenames,
              name,
              MutationModel = "Stepwise",
              MutationRate  = 0,
              MutationRange = 0.5,
              MutationRate2 = 0,
              MutationMatrix,
              Stabilization = "None",
              MaxStabilizedMutrate = 1,
              femaleMutationModel,
              femaleMutationRate,
              femaleMutationRange,
              femaleMutationRate2,
              femaleMutationMatrix,
              maleMutationModel,
              maleMutationRate,
              maleMutationRange,
              maleMutationRate2,
              maleMutationMatrix)
```

Arguments

`frequencies`	The first input of FamiliasLocus may be either a vector containing allele frequencies, or a previously created FamiliasLocus object. In the first case, the vector may include a possible silent allele; that it is silent is indicated in the allelenames. The frequencies must sum to 1. In the second case, the new object will be identical to the old object in terms of frequencies, names of alleles, and name of locus, so the 'allelenames' and 'name' parameters must be missing. However, at least one Mutation parameter or the Stabilization parameter must be non-missing, and new mutation matrices will be constructed based on these. If all Mutation parameters are missing, stabilized mutation matrices will be produced based on the mutation matrices of the old object.
`allelenames`	Names of the alleles, like `15` or 'A'. Note that the last allele may be called 'Silent' (or 'silent'). It is then treated as a silent allele in subsequent likelihood calculations. The default is to use the names attribute of the frequencies, if it exists; otherwise the default is to use consecutive integers, starting at 1. Note that if the 'Stepwise' mutation model is used, allele names (except for a silent allele) must be integers, with microvariants named as for example 15.2.
`name`	Characters like 'D3S1358', used to identify the locus (marker). The default is to use the name of the frequencies argument to this function.
`MutationModel`	The mutation model, used to create the mutation matrix. It may be 'Equal', 'Proportional', 'Stepwise', or 'Custom', see Details.
`MutationRate`	The mutation rate; for the 'Stepwise' model the rate of integer-step mutations. It is not used when the MutationModel is 'Custom'.
`MutationRange`	Only used when the MutationModel is 'Stepwise'. It then indicates the relative probability of mutating n+1 steps versus mutating n steps.
`MutationRate2`	Only used when the MutationModel is 'Stepwise'. It then indicates the rate of non-integer-step mutations, e.g., mutations from an allele with an integer name to alleles with decimal names indicating microvariants.
`MutationMatrix`	Only used when the MutationModel is 'Custom'. It then directly specifies the mutation matrix.
`Stabilization`	The possible values are 'None', 'DP', 'RM', and 'PM', with 'None' being the default. The other values adjust the mutation matrices so that allele frequencies after one or more generations of mutations will be equal to the original allele frequencies. See Details.

`MaxStabilizedMutrate`
: Not used when stabilization is 'None'. Otherwise it indicates an upper bound for the specific mutation rate for each allele allowed in the mutation matrices after stabilization.

`femaleMutationModel`
: Specifies a separate female value for MutationModel; defaults to MutationModel.

`femaleMutationRate`
: Specifies a separate female value for MutationRate; defaults to MutationRate.

`femaleMutationRange`
: Specifies a separate female value for MutationRange; defaults to MutationRange.

`femaleMutationRate2`
: Specifies a separate female value for MutationRate2; defaults to MutationRate2.

`femaleMutationMatrix`
: Specifies a separate female value for MutationMatrix; defaults to MutationMatrix.

`maleMutationModel`
: Specifies a separate male value for MutationModel; defaults to MutationModel.

`maleMutationRate`
: Specifies a separate male value for MutationRate; defaults to MutationRate.

`maleMutationRange`
: Specifies a separate male value for MutationRange; defaults to MutationRange.

`maleMutationRate2`
: Specifies a separate male value for MutationRate2; defaults to MutationRate2.

`maleMutationMatrix`
: Specifies a separate male value for MutationMatrix; defaults to MutationMatrix.

Details

The probabilities for when and how mutations happen can be specified in mutation matrices, where the row corresponding to an allele indicates the probabilities that the allele is transferred as the allele indicated by the column. Mutation matrices may be specified directly in the MutationMatrix parameters by setting the value of the MutationModel parameter to 'Custom'. Otherwise they are computed based on the values of the MutationModel, MutationRate, MutationRate2, and MutationRange parameters. If MutationModel is 'Equal', there is an equal probability of mutating to any non-silent allele, given that a mutation happens. This model is referred to as 'Equal probability (simple and fast)' in Familias 2.0. If MutationModel is 'Proportional', the probability of mutating to any non-silent allele is proportional to its frequency. It is referred to as 'Probability proportional to frequency (stable)' in Familias 2.0. If MutationModel is 'Stepwise', it is required that the names of all non-silent alleles are positive integers, indicating the number of sequence repetitions of an STR marker, or decimal numbers with a single decimal, such as '15.2', indicating a microvariant. Mutations are then divided into two types: Those that add or subtract an integer to the allele, and those that add or subtract some fractional amount. The rate of these two types of mutations are given separately as MutationRate and MutationRate2, respectively. Relative probabilities of different mutatitons of the first type are specified using the MutationRange parameter. The model with only integer alleles is referred to as 'Probability decreasing with range (equal)' in Familias 2.0, while the more general model is called 'Extended stepwise' in Familias 3.0. Note that the probability of mutations to or from silent alleles is set to zero in all models except the 'Custom' model. For the mutation matrix of the 'Custom' model, the row names and column names of the matrix must coincide and be equally sorted. Furthermore, the allele names must be the same as the row names (and hence column names) of the mutation matrix and be equally sorted.

The 'Stabilization' parameter may be used to change the mutation matrices so that they become stationary relative to the frequencies vector. See the references. When the 'PM' setting is used together with the 'Stepwise' MutationModel and all allele names are integers, the resulting model is referred to as 'Probability decreasing with range (stable)' in Familias 2.0.

Value

A list of class `FamiliasLocus` containing

`locusname`	The name of the locus.
`alleles`	The frequencies of the alleles. The names of the alleles are included as the vector names.
`femaleMutationType`	A string specifying the type of the female mutations.
`femaleMutationMatrix`	The mutation matrix used for female transfer.
`maleMutationType`	A string specifying the type of the male mutations.
`maleMutationMatrix`	The mutation matrix used for male transfer.
`simpleMutationMatrices`	Indicates whether the probability of mutating to an allele is always independent of which allele the mutation happens from. If this is true, some likelihood computations can be done faster.
`Stabilization`	The stabilization method used.

References

Egeland, Kling, Mostad: Relationship Inference with Familias and R. (Academic press, 2016, https://www.familias.name/book.html). Simonsson, Mostad: Stationary Mutation models (FSI: Genetics, 2016).

Examples

```
#Simple examples
FamiliasLocus(1:4/10)
FamiliasLocus(frequencies = c(0.1, 0.2, 0.3, 0.4),
allelenames = c("A", "B", "C", "D"), name = "locus1")

#Modified to include a silent frequency
FamiliasLocus(frequencies = c(0.1, 0.2, 0.3, 0.3, 0.1),
allelenames = c("8", "9", "10", "11", "silent"), name = "locus1")

#Mutation rates added
```

```
FamiliasLocus(frequencies = c(0.1, 0.2, 0.3, 0.4),
allelenames = c("8", "9", "10", "11"), name = "locus1",
femaleMutationRate = 0.001, maleMutationRate = 0.005)

#Mutation matrices specified directly
MM <- matrix(c(0.99, 0.005, 0.003, 0.002, 0.005, 0.99, 0.005, 0,
               0, 0.005, 0.99, 0.005, 0.002, 0.003, 0.005, 0.99),
             ncol = 4, nrow = 4, byrow = TRUE)
FamiliasLocus(frequencies = c(0.1, 0.2, 0.3, 0.4),
allelenames= c("08", "09", "10", "11"), name = "locus1",
MutationModel = "Custom", MutationMatrix = MM)

#A locus is first created, and then edited
loc <- FamiliasLocus(c(0.2, 0.5, 0.3))
loc2 <- FamiliasLocus(loc, maleMutationRate = 0.001)
FamiliasLocus(loc2, Stabilization = "PM")

#A locus using standard Norwegian frequencies is created
data(NorwegianFrequencies)
FamiliasLocus(NorwegianFrequencies$TH01)
```

`FamiliasPedigree` *Creates an object storing a pedigree*

Description

Creates and stores an object containing a pedigree in much the same way as the 'pedigree' function of the 'kinship2' package. It is checked that the input represents a correct pedigree. The main differences is that a person is allowed to have one parent present and one absent in the pedigree. Another difference is that no disease parameters are included. The result is an object with class `FamiliasPedigree`.

Usage

```
FamiliasPedigree(id, dadid, momid, sex)
```

Arguments

`id`	A vector containing unique identifiers of all individuals in the pedigree.
`dadid`	Indicates the fathers of individuals. The vector must have the same length as the id vector and contain either values from it, indicating that the individual with this position in the id vector has the given father, or NA.
`momid`	Indicates the mothers of individuals. The vector must have the same length as the id vector and contain either values from it, indicating that the individual with this position in the id vector has the given mother, or NA.
`sex`	A vector of the same length as the id vector, indicating the gender of individuals. Values must be either "female" or "male".

Details

The objects created by the `FamiliasPedigree` function represent both a simplification and a generalization of the objects generated by the 'pedigree' function of the 'kinship2' package. It is a simplification in that parameters concerning disease are dropped, but it is a generalization in the sense that persons are allowed to have exactly one parent present in the pedigree. This generalization is necessary for the Familias package, as results from the `FamiliasPosterior` function may change when a single ancestor (father or mother) is added to a single person in a pedigree. Such changes may occur when a non-zero mutation rate is used together with a non-stable mutation model. The `FamiliasPosterior` and `FamiliasPrior` functions can use either pedigree type as input.

Value

A list of class `FamiliasPedigree` containing

`id`	The same vector as the id input.
`findex`	A vector of indices of fathers of persons. Zero indicates that the person has no father in the pedigree.
`mindex`	A vector of indices of mothers of persons. Zero indicates that the person has no mother in the pedigree.
`sex`	The same vector as the sex input.

Author(s)

Petter Mostad mostad@chalmers.se

Examples

```
#A nuclear family of three:
ped <- FamiliasPedigree(c("mother", "father", "child"),
                        c(NA, NA, "father"),
                        c(NA, NA, "mother"),
                        c("female", "male", "female"))
plot(ped, symbolsize = 2, cex = 2, family = "mono")

#Generating the two pedigrees needed for a traditional paternity case
ped1 <- FamiliasPedigree(c("mother", "child", "AF"),
                         c(NA, NA, NA),
                         c(NA, "mother", NA),
                         c("female", "female", "male"))
ped2 <- FamiliasPedigree(c("mother", "child", "AF"),
                         c(NA, "AF", NA),
                         c(NA, "mother", NA),
                         c("female", "female", "male"))

#Generating the two pedigrees needed for a duo case
ped1 <- FamiliasPedigree(c("child", "AF"), c(NA, NA), c(NA, NA), c("male", "male"))
ped2 <- FamiliasPedigree(c("child", "AF"), c("AF", NA), c(NA, NA), c("male", "male"))
```

`FamiliasPosterior` *Calculates posterior probabilities and likelihoods for pedigrees*

Description

The calculations of the windows version of Familias 2.0 are made available in an R environment.

Usage

```
FamiliasPosterior(pedigrees, loci, datamatrix, prior, ref = 1, kinship = 0,
                  simplifyMutations = FALSE)
```

Arguments

`pedigrees`	An object of type 'FamiliasPedigree' or 'pedigree', or a list of such objects.
`loci`	A FamiliasLocus object or a list of such objects.
`datamatrix`	A data frame. The row names must be the names of the persons you have data for. The columns contain the alleles, two columns for each marker, in the same order used in the loci list.
`prior`	A vector of non-negative probabilities summing to 1. As default a flat prior is used, with all values equal.
`ref`	The index in the list of pedigrees of the pedigree which should be used as reference when computing likelihood ratios. The default value is 1.
`kinship`	A real in [0,1], commonly denoted theta in forensics included to model subpopulation effects as departures from Hardy-Weinberg equilibrium. The default value is zero.
`simplifyMutations`	In pedigrees with several generations multistep mutations may happen. If the probability of mutating to an allele depends on which allele it mutates from, exact likelihood computations must keep track of all the possible values of mutated alleles in such multistage mutations, and this may slow down computations considerably. Instead, one may use in computations for multistage mutations a single allele that is not among those observed in the data. When this approach gives exact results it is always used; in other cases one may choose to use it as an approximation by setting simplifyMutations to TRUE. The properties of the extra allele is the weighted average (by population frequencies) of the alleles not observed in the data.

Value

`posterior`	Probabilities for each pedigree.
`prior`	Prior returned for convenience and backward compatibility.
`LR`	Likelihood ratios using the pedigree indicated with the ref parameter (default value 1) as basis.

`LRperMarker`	Likelihood ratios per marker using the pedigree indicated with the ref parameter (default value 1) as basis.
`likelihoods`	For each pedigree.
`likelihoodsPerSystem`	Likelihoods for each locus and each pedigree.

Author(s)

Petter Mostad <mostad@chalmers.se> and Thore Egeland <Thore.Egeland@nmbu.com>.

Examples

```
#Example 1
#Data is available for "mother", "daughter" and "AF" for two loci (systems).
#Three pedigrees are defined, with "mother" being the mother of "daughter"
#in all cases. "AF" may be the father (ped1), unrelated (ped1) or
#uncle (ped3). The posterior probabilities for the pedigrees are calculated
#and likelihoods are also given so that likelihood ratios can be computed.
#Compared to the windows version of Familias it is easy to plot pedigrees and
#define arbitrary priors for the three alternative pedigrees.
#The implementation uses the R package kinship2 to define and plot pedigrees.

persons <- c("mother", "daughter", "AF")
ped1 <- FamiliasPedigree(id = persons,
                         dadid = c(NA, "AF", NA),
                         momid = c(NA, "mother", NA),
                         sex = c("female", "female", "male"))
ped2 <- FamiliasPedigree(id = c(persons, "TF"),
                         dadid = c(NA, "TF", NA, NA),
                         momid = c(NA, "mother", NA, NA),
                         sex = c("female", "female", "male", "male"))
ped3 <- FamiliasPedigree(id = c(persons, "TF", "gf", "gm"),
                         dadid = c(NA, "TF", "gf", "gf", NA, NA),
                         momid = c(NA, "mother", "gm", "gm", NA, NA),
                         sex = c("female", "female", "male", "male", "male", "female"))
op <- par(mfrow = c(1,3))
plot(ped1); title("ped1: AF is father")
plot(ped2); title("ped2: AF is unrelated")
plot(ped3); title("ped3: AF is uncle")
par(op)

mypedigrees <- list(isFather = ped1, unrelated = ped2, isUncle = ped3)

locus1 <- FamiliasLocus(frequencies = c(0.1, 0.2, 0.3, 0.4),
                        allelenames = c("A", "B", "C", "D"),
                        name = "locus1")
locus2 <- FamiliasLocus(c(0.2, 0.3, 0.5),
                        c(17, 18, 19),
                        "loc2",
                        femaleMutationRate = 0.05)
myloci <- list(locus1, locus2)
```

```
datamatrix <- data.frame(locus1.1 = c("A", "A", "A"),
                         locus1.2 = c ("B", "B", "C"),
                         locus2.1 = c(17, 19, 19),
                         locus2.2 = c(18, 18, 18))
rownames(datamatrix) <- persons

## Not run:
result <- FamiliasPosterior(mypedigrees, myloci, datamatrix, ref = 2)

## End(Not run)



#Example 2: Using FamiliasPedigree
persons <- c("person", "AF")
sex <- c("male", "male")
ped1 <- FamiliasPedigree(id = persons, dadid = c(NA, NA), momid = c(NA, NA), sex = sex)
ped2 <- FamiliasPedigree(id = persons, dadid = c("AF", NA), momid = c(NA, NA), sex = sex)
mypedigrees <- list(unrelated = ped1, isFather = ped2)

locus1 <- FamiliasLocus(c(0.1, 0.2, 0.3, 0.4), c("A", "B", "C", "D"), "locus1",
                        maleMutationModel = "Equal", maleMutationRate = 0.005)
locus2 <- FamiliasLocus(c(0.2, 0.3, 0.5), c(17, 18, 19), "locus2",
                        maleMutationModel = "Equal", maleMutationRate = 0.005)
myloci <- list(locus1, locus2)

datamatrix <- data.frame(locus1.1 = c("A", "A"), locus1.2 = c("B", "B"),
                         locus2.1 = c(17, 19),   locus2.2 = c(18, 18))
rownames(datamatrix) <- persons
## Not run:
result <- FamiliasPosterior(mypedigrees, myloci, datamatrix)

## End(Not run)

#Example 3: User-specified mutation matrices
persons <- c("son", "mother", "AF")
sex <- c("male", "female", "male")
ped1 <- FamiliasPedigree(id = persons, dadid = c(NA, NA, NA),
                         momid = c("mother", NA, NA), sex = sex)
ped2 <- FamiliasPedigree(id = persons, dadid = c("AF", NA, NA),
                         momid = c("mother", NA, NA), sex = sex)
mypedigrees <- list(unrelated = ped1, isFather = ped2)

mut = matrix(c(0.99, 0.005, 0.003, 0.002,
               0.004, 0.99, 0.004, 0.002,
               0.002, 0.004, 0.99, 0.004,
               0.002, 0.003, 0.005, 0.99),
             4, 4, byrow = TRUE)
locus1 <- FamiliasLocus(c(0.1, 0.2, 0.3, 0.4), c("A", "B", "C", "D"), "locus1",
                        maleMutationModel = "Custom",
                        maleMutationMatrix = mut,
                        femaleMutationModel = "Custom",
                        femaleMutationMatrix = mut)
```

```
datamatrix <- data.frame(locus1.1 = c("A", "A", "C"),
                         locus1.2 = c("B", "A", "C"))
rownames(datamatrix) <- persons
## Not run:
result <- FamiliasPosterior(mypedigrees, locus1, datamatrix)

## End(Not run)

#Example 4: Adding an untyped ancestor affects the likelihood
#when subpopulation correction is used even if the mutation matrix is stationary

# ped1 is son by himself
## Not run:
ped1 <- FamiliasPedigree(id = "son",
                         dadid = c(NA),
                         momid = c(NA),
                         sex = "male")

# ped2 describes son with parents
persons <- c("son", "mother", "AF")
sex <- c("male", "female", "male")
ped2 <- FamiliasPedigree(id = persons,
                         dadid = c("AF", NA, NA),
                         momid = c("mother", NA, NA),
                         sex = sex)

# only the son is typed
datamatrix <- matrix(c("A", "A"), ncol = 2)
rownames(datamatrix) <- "son"

f <- setNames(c(0.3, 0.7), c("A", "B"))

locus1 <- FamiliasLocus(f,
                        MutationModel = "Proportional",
                        MutationRate = 1e-2)

# if Fst=0, then ped1 and ped2 are equivalent
pr_ped1_HW <- FamiliasPosterior(list(ped1), locus1, datamatrix)$likelihoods
pr_ped2_HW <- FamiliasPosterior(list(ped2), locus1, datamatrix)$likelihoods

# verify calculations by hand
stopifnot(all.equal(pr_ped1_HW, pr_ped2_HW))
stopifnot(all.equal(pr_ped1_HW, as.numeric(f["A"]^2)))
stopifnot(all.equal(pr_ped2_HW, as.numeric(f["A"]^2)))

# if Fst > 0, then ped1 and ped2 are not equivalent
Fst <- 0.03

pr_ped1_Fst <- FamiliasPosterior(list(ped1), locus1, datamatrix, kinship = Fst)$likelihoods
pr_ped2_Fst <- FamiliasPosterior(list(ped2), locus1, datamatrix, kinship = Fst)$likelihoods

# pr_ped1_Fst and pr_ped2_Fst should not be equal: stop if equal
```

```
stopifnot(!isTRUE(all.equal(pr_ped1_Fst, pr_ped2_Fst)))

# compute pr_ped1_Fst by hand to verify the result
pr_ped1_Fst_manual <- as.numeric(f["A"]*Fst + f["A"]^2 * (1-Fst))
stopifnot(all.equal(pr_ped1_Fst, pr_ped1_Fst_manual))

# compute pr_ped2_Fst by hand to verify the result
# computing Pr(son=A,A|ped2) with subpopulation correction and mutations
# requires an explicit sum over the two parental alleles

pr_AA <- as.numeric(f["A"] * Fst + f["A"]^2 * (1-Fst))
pr_AB <- 2 * as.numeric(f["A"] *f ["B"] * (1-Fst))
pr_BB <- as.numeric(f["B"]*Fst + f["B"]^2 * (1-Fst))

M <- locus1$maleMutationMatrix

pr_AA_given_AA <- M["A","A"]^2
pr_AA_given_AB <- M["A","A"] * M["B","A"]
pr_AA_given_BB <- M["B","A"]^2

pr_ped2_Fst_manual <- pr_AA*pr_AA_given_AA + pr_AB*pr_AA_given_AB + pr_BB*pr_AA_given_BB

stopifnot(all.equal(pr_ped2_Fst, pr_ped2_Fst_manual))

## End(Not run)
```

FamiliasPrior *Calculates a prior distribution for a list of pedigrees*

Description

By default the same prior probability is assigned to all pedigrees in the list, but this can be adjusted with the function parameters. It is computed which persons are common to all the pedigrees listed, and they are handled in a special way: It is with relation to these persons that the number of generations and other parameters are computed. Also, the function will search for and remove pedigrees that are "equivalent" in terms of representing the relationship between these core persons. So if another pedigree is added, with all new persons, the function will return with an error message.

Usage

```
FamiliasPrior(pedigrees,
              generationsParameter = 1,
              inbreedingParameter = 1,
              partnerParameter = 1,
              maxGenerations)
```

Arguments

`pedigrees`	A list of objects of class either 'pedigree' or 'FamiliasPedigree'.
`generationsParameter`	Non-negative real. A value of 1 indicates no influence of the parameter. A value above 1 (below 1) increases (decreases) the prior probability for pedigrees with many generations.
`inbreedingParameter`	Non-negative real. A pedigree is inbred if parents are related within the pedigree. If 0, all inbred pedigrees are assigned a prior probability 0. A value of 1 indicates no influence of inbreeding. A value above 1 (below 1) increases (decreases) the prior for inbred pedigrees.
`partnerParameter`	Non-negative real (previously referred to as promiscuity parameter). If 0, all pedigrees where parents have children by different partners, are assigned prior 0. A value of 1 indicates no influence of the parameter. A value above 1 (below 1) increases (decreases) the prior for partners having children by different partners.
`maxGenerations`	Integer giving the maximum number of generations; pedigrees with more generations than this are assigned a zero prior probability.

Details

See https://familias.name/manual.pdf for a complete description of the parametric models implemented.

Value

The prior, i.e., a vector of real numbers summing to 1.

Author(s)

Petter Mostad <mostad@chalmers.se> and Thore Egeland <Thore.Egeland@nmbu.com>

Examples

```
persons <- c("mother", "daughter", "AF")
ped1 <- FamiliasPedigree(id = persons,
                         dadid = c(NA, "AF", NA),
                         momid = c(NA, "mother", NA),
                         sex = c("female", "female", "male"))
ped2 <- FamiliasPedigree(id = c(persons, "TF"),
                         dadid = c(NA, "TF", NA, NA),
                         momid = c(NA, "mother", NA, NA),
                         sex = c("female", "female", "male", "male"))
ped3 <- FamiliasPedigree(id = c(persons, "TF", "gf", "gm"),
                         dadid = c(NA, "TF", "gf", "gf", NA, NA),
                         momid = c(NA, "mother", "gm", "gm", NA, NA),
                         sex = c("female", "female", "male", "male", "male", "female"))

mypedigrees <- list(isFather = ped1, unrelated = ped2, isUncle = ped3)
```

```
FamiliasPrior(mypedigrees)

granddad <- FamiliasPedigree(id = c(persons, "TF", "gm"),
                             dadid = c(NA, "TF", NA, "AF", NA),
                             momid = c(NA, "mother", NA, "gm", NA),
                             sex = c("female", "female", "male", "male", "female"))
FamiliasPrior(c(mypedigrees, list(granddad)))
FamiliasPrior(c(mypedigrees, list(granddad)), maxGenerations = 1)
```

`NorwegianFrequencies` *A list of markers with allele names and frequencies*

Description

The information represents allele frequencies for a range of different markers based on a data base in Norway.

Usage

```
data(NorwegianFrequencies)
```

Format

A list with named components, with names corresponding to the names of the loci. Each component is a vector of frequencies, with the names attribute of the vector equal to the names of the alleles.

Source

Norwegian Institute of Public Health, Department of Family Genetics: *Dupuy et al: Frequency data for 35 autosomal STR markers in a Norwegian, an East African, and East Asian and Middle Asian population and simulation of adequate database size. Forensic Science International: Genetics Supplement Series, Volume 4, issue 1, 2013*

Examples

```
data(NorwegianFrequencies)

#Displaying the Norwegian frequencies of the
NorwegianFrequencies$TPOX

#Generating a FamiliasLocus with these frequencies
FamiliasLocus(NorwegianFrequencies$TPOX)

#Including a non-zero male mutation rate
FamiliasLocus(NorwegianFrequencies$TPOX, maleMutationRate = 0.005)

#Listing the names of available markers
names(NorwegianFrequencies)
```

plot.FamiliasPedigree *Plotting function for FamiliasPedigree objects*

Description

The function piggybacks the plot function for pedigree objects from the kinship2 package to create a plotting function for FamiliasPedigree objects. Before conversion to a pedigree object, additional parents are added to the pedigree so that each person has either zero or two parents within the pedigree.

Usage

```
## S3 method for class 'FamiliasPedigree'
plot(x, y, ...)
```

Arguments

x	An object of class FamiliasPedigree.
y	Not used in this printing function.
...	Other arguments transferred to ‘kinship2::plot.pedigree()‘.

Details

Graphical parameters used in ‘kinship2::plot.pedigree()‘ may be input via ‘plot.FamiliasPedigree()‘.

Value

A plot is produced.

Author(s)

Petter Mostad mostad@chalmers.se

Examples

```
ped <- FamiliasPedigree(id = c("child", "AF"),
                        momid = c(NA, NA),
                        dadid = c("AF", NA),
                        sex = c("male", "male"))
plot(ped)
dev.new()
plot(ped, symbolsize = 2, cex = 2, family = "mono")
```

Index