Package 'optiSel'

July 22, 2025

Type Package

Title Optimum Contribution Selection and Population Genetics

Version 2.0.9

Date 2024-07-08

Depends R (>= 3.5.0)

Description A framework for the optimization of breeding programs via optimum contribution selection and mate allocation. An easy to use set of function for computation of optimum contributions of selection candidates, and of the population genetic parameters to be optimized. These parameters can be estimated using pedigree or genotype information, and include kinships, kinships at native haplotype segments, and breed composition of crossbred individuals. They are suitable for managing genetic diversity, removing introgressed genetic material, and accelerating genetic gain. Additionally, functions are provided for computing genetic contributions from ancestors, inbreeding coefficients, the native effective size, the native genome equivalent, pedigree completeness, and for preparing and plotting pedigrees. The methods are described in:\n Wellmann, R., and Pfeiffer, I. (2009) <doi:10.1017/S0016672309000202>.\n Wellmann, R., and Bennewitz, J. (2011) <doi:10.2527/jas.2010-3709>.\n Wellmann, R., Hartwig, S., Bennewitz, J. (2012) <doi:10.1186/1297-9686-44-34>.\n de Cara, M. A. R., Villanueva, B., Toro, M. A., Fernandez, J. (2013) <doi:10.1111/mec.12560>.\n Wellmann, R., Bennewitz, J., Meuwissen, T.H.E. (2014) <doi:10.1017/S0016672314000196>.\n Wellmann, R. (2019) <doi:10.1186/s12859-018-2450-5>.

License GPL-2

Imports Matrix, plyr, kinship2, nadiv, pedigree, pspline, stringr, MASS, methods, stats, purrr, graphics, quadprog, data.table, magic, parallel, doParallel, foreach, ECOSolveR, reshape2, optiSolve, Rcpp (>= 0.12.4)

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, ggplot2, rmarkdown, alabama, cccp, nloptr, Rsymphony, smacof

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-07-08 12:20:02 UTC

Author Robin Wellmann [aut, cre]

Maintainer Robin Wellmann <r.wellmann@uni-hohenheim.de>

Contents

`optiSel-package` *Optimum Contribution Selection and Population Genetics*

Description

A framework for the optimization of breeding programs via optimum contribution selection and mate allocation. An easy to use set of function for computation of optimum contributions of selection candidates, and of the population genetic parameters to be optimized. These parameters can be estimated using pedigree or genotype information, and include kinships, kinships at native haplotype segments, and breed composition of crossbred individuals. They are suitable for managing genetic diversity, removing introgressed genetic material, and accelerating genetic gain. Additionally, functions are provided for computing genetic contributions from ancestors, inbreeding coefficients, the native effective size, the native genome equivalent, pedigree completeness, and for preparing and plotting pedigrees. The methods are described in:\n Wellmann, R., and Pfeiffer, I. (2009) <doi:10.1017/S0016672309000202>.\n Wellmann, R., and Bennewitz, J. (2011) <doi:10.2527/jas.2010-3709>.\n Wellmann, R., Hartwig, S., Bennewitz, J. (2012) <doi:10.1186/1297-9686-44-34>.\n de Cara, M. A. R., Villanueva, B., Toro, M. A., Fernandez, J. (2013) <doi:10.1111/mec.12560>.\n Wellmann, R., Bennewitz, J., Meuwissen, T.H.E. (2014) <doi:10.1017/S0016672314000196>.\n Wellmann, R. (2019) <doi:10.1186/s12859-018-2450-5>.

Details

Optimum Contribution Selection

After kinships, breeding values and/or native contributions of the selection candidates have been computed, function candes can be used to create an R-object containing all this information. The current average kinships and trait values are estimated by this function, and the available objective functions and constraints for optimum contribution selection are reported. The following function can then be used to compute optimum contributions:

opticont	Calculates optimum genetic contributions of selection candidates to the next generation, and checks if all constraints are fulfilled.

Function noffspring can be used to compute the optimum numbers of offspring of selection candidates from their optimum contributions. Function matings can be used for mate allocation.

Kinships

For pairs of individuals the following kinships can be computed:

pedIBD	Calculates **ped**igree based probability of alleles to be **IBD** ("pedigree based kinship""),
segIBD	Calculates **seg**ment based probability of alleles to be **IBD** ("segment based kinship"),
pedIBDatN	Calculates **ped**igree based probability of alleles to be **IBD at** segments with **N**ative origin,
segIBDatN	Calculates **seg**ment based probability of alleles to be **IBD at** segments with **N**ative origin,
pedIBDorM	Calculates **ped**igree based probability of alleles to be **IBD or M**igrant alleles,
segIBDandN	Calculates **seg**ment based probability of alleles to be **IBD and** have **N**ative origin,

segN	Calculates **seg**ment based probability of alleles to have **N**ative origin,
makeA	Calculates the pedigree-based additive relationship matrix.

Phenotypes and results from these functions can be combined with function candes into a single R object, which can then be used as an argument to function opticont.

The segment based kinship can be used to calculate the optimum contributions of different breeds to a hypothetical multi-breed population with maximum genetic diversity by using function opticomp.

Function sim2dis can be used to convert a similarity matrix (e.g. a kinship matrix) into a dissimilarity matrix which is suitable for multidimensional scaling.

Breed Composition

The breed composition of crossbred individuals can be accessed with

pedBreedComp	Calculates **ped**igree based the **Breed Comp**osition, which is the genetic contribution of each individual from other breeds and from native founders. The native contribution is the proportion of the genome not originating from other breeds.
segBreedComp	Calculates **seg**ment based the **Breed Comp**osition. The native contribution is the proportion of the genome belonging to segments that have low frequency in other breeds.

The native contributions obtained by the above functions can be constrained or maximized with function opticont to remove introgressed genetic material, or alternatively, the segment-based native contribution can be considered a quantitative trait and included in a selection index.

Haplotype frequencies

Frequencies of haplotype segments in particular breeds can be computed and plotted with

haplofreq	Calculates the maximum frequency each segment has in a set of reference breeds, and the name of the breed in which the segment has maximum frequency. Identification of native segments.
freqlist	Combines results obtained with function haplofreq for different reference breeds into a single R object which is suitable for plotting.
plot.HaploFreq	Plots frequencies of haplotype segments in particular reference breeds.

Inbreeding Coefficients and Genetic Contributions

The inbreeding coefficients and genetic contributions from ancestors can be computed with:

pedInbreeding	Calculates **ped**igree based **Inbreeding**.
segInbreeding	Calculates **seg**ment based **Inbreeding**, i.e. inbreeding based on runs of homozygosity (ROH).
genecont	Calculates **gene**tic **cont**ributions each individual has from all it's ancestors in the pedigree.

Preparing and plotting pedigree data

There are some functions for preparing and plotting pedigree data

prePed	**pre**pares a **Ped**igree by sorting, adding founders and pruning the pedigree,
completeness	Calculates pedigree **completeness** in all ancestral generations,
summary.Pedig	Calculates number of equivalent complete generations, number of fully traced generations, number of maximum generations traced, index of pedigree completeness, inbreeding coefficients,
subPed	Creates a **sub**set of a large **Ped**igree,
pedplot	Plots a pedigree,
sampleIndiv	Samples individuals from a pedigree.

Population Parameters

Finally, there are some functions for estimating population parameters:

conttac	Calculates genetic **cont**ributions of breeds **t**o **a**ge **c**ohorts,
summary.candes	Calculates for every age cohort several genetic parameters. These may include average kinships, kinships at native loci, the native effective size, and the native genome equivalent.

Genotype File Format

All functions reading genotype data assume that the files are in the following format:

Genotypes are phased and missing genotypes have been imputed. Each file has a header and no row names. Cells are separated by blank spaces. The number of rows is equal to the number of markers from the respective chromosome and the markers are in the same order as in the `map`. There can be some extra columns on the left hand side containing no genotype data. The remaining columns contain genotypes of individuals written as two alleles separated by a character, e.g. A/B, 0/1, A|B, A B, or 0 1. The same two symbols must be used for all markers. Column names are the IDs of the individuals. If the blank space is used as separator then the ID of each individual should be repeated in the header to get a regular delimited file. The columns to be skipped and the individual IDs must have no white spaces.

Use function read.indiv to extract the IDs of the individuals from a genotype file.

Author(s)

Robin Wellmann [aut, cre]

Maintainer: Robin Wellmann <r.wellmann@uni-hohenheim.de>

References

de Cara MAR, Villanueva B, Toro MA, Fernandez J (2013). Using genomic tools to maintain diversity and fitness in conservation programmes. Molecular Ecology. 22: 6091-6099

Wellmann, R., and Pfeiffer, I. (2009). Pedigree analysis for conservation of genetic diversity and purging. Genet. Res. 91: 209-219

Wellmann, R., and Bennewitz, J. (2011). Identification and characterization of hierarchical structures in dog breeding schemes, a novel method applied to the Norfolk Terrier. Journal of Animal Science. 89: 3846-3858

Wellmann, R., Hartwig, S., Bennewitz, J. (2012). Optimum contribution selection for conserved populations with historic migration; with application to rare cattle breeds. Genetics Selection Evolution. 44: 34

Wellmann, R., Bennewitz, J., Meuwissen, T.H.E. (2014) A unified approach to characterize and conserve adaptive and neutral genetic diversity in subdivided populations. Genet Res (Camb). 69: e16

Wellmann, R. (2019). Optimum Contribution Selection and Mate Allocation for Breeding: The R Package optiSel. BMC Bioinformatics 20:25

Examples

```
#See ?opticont for optimum contribution selection
#These examples demonstrate computation of some population genetic parameters.

data(ExamplePed)
Pedig <- prePed(ExamplePed, thisBreed="Hinterwaelder", lastNative=1970)
head(Pedig)

##########################################
# Evaluation of                          #
#    - kinships                          #
#    - genetic diversities               #
#    - native effective size             #
#    - native genome equivalent          #
##########################################

phen    <- Pedig[Pedig$Breed=="Hinterwaelder",]
pKin    <- pedIBD(Pedig)
pKinatN <- pedIBDatN(Pedig, thisBreed="Hinterwaelder")
pop     <- candes(phen=phen, pKin=pKin, pKinatN=pKinatN, quiet=TRUE, reduce.data=FALSE)
Param   <- summary(pop, tlim=c(1970,2005), histNe=150, base=1800, df=4)


plot(Param$t, Param$Ne, type="l", ylim=c(0,150),
     main="Native Effective Size", ylab="Ne", xlab="")

matplot(Param$t, Param[,c("pKin", "pKinatN")],
        type="l",ylim=c(0,1),main="Kinships", xlab="Year", ylab="mean Kinship")
abline(0,0)
legend("topleft", legend = c("pKin", "pKinatN"), lty=1:2, col=1:2, cex=0.6)

info <- paste("Base Year =", attributes(Param)$base, "  historic Ne =", attributes(Param)$histNe)

plot(Param$t,Param$NGE,type="l",main="Native Genome Equivalents",
     ylab="NGE",xlab="",ylim=c(0,7))
mtext(info, cex=0.7)
```

```
##########################################
# Genetic contributions from other breeds  #
##########################################

cont <- pedBreedComp(Pedig, thisBreed='Hinterwaelder')
contByYear <- conttac(cont, Pedig$Born, use=Pedig$Breed=="Hinterwaelder", mincont=0.04, long=FALSE)
round(contByYear,2)

barplot(contByYear, ylim=c(0,1), col=1:10, ylab="genetic contribution",
        legend=TRUE, args.legend=list(x="topleft",cex=0.6))


#####################################################
# Frequencies of haplotype segments in other breeds  #
#####################################################

data(map)
data(Cattle)
dir   <- system.file("extdata", package="optiSel")
files <- file.path(dir, paste("Chr", 1:2, ".phased", sep=""))

Freq <- freqlist(
  haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="Rotbunt",   minSNP=20),
  haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="Holstein",  minSNP=20),
  haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="Fleckvieh", minSNP=20)
  )

plot(Freq, ID=1, hap=2, refBreed="Rotbunt")
```

`agecont` *Contributions of age cohorts to the population*

Description

Contributions of age classes to the population are calculated such that the contribution of each age class to the population is proportional to the expected proportion of offspring that is not yet born.

Note that the contribution of a class to the population is not equal to the proportion of individuals belonging to the class.

Usage

```
agecont(Pedig, use=Pedig$Born >= quantile(Pedig$Born, 0.75), maxAge=NA)
```

Arguments

`Pedig`	Pedigree with colums `Indiv`, `Sire`, `Dam`, and `Born`, usually created with function prePed.

use	Logical vector or character vector with IDs indicating the individuals from the current population.
maxAge	Parents that are more than maxAge years older than their offspring are ignored. By default, old parents are not ignored.

Details

Contributions of age classes to the population are calculated such that the contribution of each age class to the population is proportional to the expected proportion of offspring that is not yet born.

More precisely:

Individuals born in the current year are in age class k=1. Typically, each age class spans one year. No individual can have offspring in the same age class. Males and females that are not born in the current year are assumed to have equal contributions to the population. Moreover, as stated above, it is assumed that the contribution of each class to the population is proportional to the proportion of offspring from this class that is not yet born when the individuals leaves the class.

This approach to define contributions has the advantage that it does not need to be known which individuals are still alive and which are removed from the breeding pool. Moreover, it causes old age classes to have a smaller contribution to the population than young age classes.

The contributions are estimated from the ages of the parents when the individuals in vector use were born. Obviously, the contributions of age classes to the offspring in the next year do not coincide with the contributions of the age classes to the population.

Value

Data frame containing the contributions of all age cohorts to the current population.

Examples

```
data(PedigWithErrors)
Pedig <- prePed(PedigWithErrors)
use   <- Pedig$Breed=="Hinterwaelder" & !is.na(Pedig$Born)
use   <- use & Pedig$Born>=2000 & Pedig$Born<=2004

# Calculate the contribution of each age class ##

cont <- agecont(Pedig, use)

# Contribution of each age class to
# the current population:

head(cont)

# Note: In this case, young males have a higher contribution to the
# population than young females because they are used for breeding
# for a shorter time span, i.e. they are culled earlier.

# Males and females (excluding the newborn individuals)
# have equal contributions to the current population:
```

```
sum(cont$male[-1])
#[1] 0.3925894

sum(cont$female[-1])
#[1] 0.3925894


# The total contribution of classes to the curent population is equal to 1

sum(cont$female) + sum(cont$male)
#[1] 1

# When used for OCS, the contribution of the offspring to the
# population in the next year is equal to the contribution of the individuals
# born in this year to the current population:

cont$male[1]+cont$female[1]
#[1] 0.2148212

# This is approximately 1/L, where L is the generation interval.
```

---

candes *Candidate Description*

Description

An R-Object is created containing all information describing the individuals, which is usually a sample from the current population and includes the selection candidates. Average kinships and trait values, and the available objective functions and constraints for optimum contribution selection (OCS) are reported.

Usage

```
candes(phen, cont=NULL, N=1000, quiet=FALSE, t=NA, bc=NULL, reduce.data=TRUE, ...)
```

Arguments

phen	Data frame with column `Indiv` containing animal IDs and possibly `Sex` containng sexes, coded as `'male'` and `'female'`, or `NA` if sexes are to be ignored. It also contains column `Born` with year of birth if generations are assumed to be overlapping. The other columns may contain traits, e.g. breeding values or native contributions, column `Breed` with breed names for multi-breed evaluations, logical column `isCandidate` indicating the selection candidates, and columns `Sire` and `Dam` with IDs of sires and dams.
cont	Data frame frame with column `age` (equal to the row number), and columns `male`, and `female`, containing the contributions of males and females from each age class to the population. It is usually created with function agecont. The default means that non-overlapping generations are assumed, so there is only one age class for males and one for females.

N	The population size. A small value accelerates the increase in kinship due to genetic drift. For overlapping generations it can be calculated as N=N0/r0, where N0 is the number of individuals born each year, and r0<=1 is the percentage which this age class represents in the population. The default is N=1000.
quiet	Should the report be suppressed?
t	The time at which the population should be evaluated. The default means that t=max(floor(phen$Born)).
bc	Only needed if multi-breed data is provided. Named vector with breed contributions, with component names being the names of the breeds in phen. It contains the proportion of each breed to a hypothetical multi-breed population for which the diversity across breeds should be managed. Alternatively, bc can be a character string containig the name of a kinship. In this case, optimum contributions of the breeds are determined automatically so that the mean kinship across breeds is minimized.
reduce.data	Logical. Should data from individuals not contributing to the population at time t be removed from the output?
...	One or more objects of class 'matrix', 'quadFun', or 'ratioFun' defining the pairwise kinships and native kinships of individuals.

Details

An R-Object is created containing all information describing the individuals, which is usually the current population and includes the selection candidates. Average kinships and trait values are estimated and reported. The weights of Age x Sex classes are in accordance with argument cont. The available objective functions and constraints for optimum contribution selection are reported.

Value

List of class candes with the following components:

kinship	Objects of class 'quadFun', or 'ratioFun', one for each additional parameter. These objects define the functions needed to estimate the mean kinships and mean native kinships in the next year or generation.
phen	Supplied data frame phen containing phenotypes, individual IDs, and some appended columns that are needed for OCS. These are * Column Age with the ages of the individuals, * Column Class with the Breed x Age x Sex or Breed x Age classes to which the individuals belong. * Column c0 containing the contribution each individual itself has to the current population. * Column c1 containing the contribution each individual itself has to the population in the next year (for overlapping generations) or to the next generation (for non-overlapping generations). In the latter case, c1 contains zeros. * Column isCandidate indicating the selection candidates.
mean	Data frame containing estimates of the current mean values (at time t) of the parameters in a population consisting of N individuals for which the individuals in argument phen are representative.

current	Data frame containing the same values as component mean, but also some additional information on the parameters.
bc	Character vector with optimum breed contributions (see above).
classes	Data frame containing the number of individuals in each class (column n), the contribution of each class to the population in this year/generation (column rcont0) and in the next year/generation (column rcont1), and the expected proportion of offspring animals from a given sex have at a particular age.
breed	List describing the breeds included in the data set.

Author(s)

Robin Wellmann

Examples

```
data(PedigWithErrors)

Pedig      <- prePed(PedigWithErrors, thisBreed="Hinterwaelder", lastNative=1970,
                     keep=PedigWithErrors$Born%in%1992)
use        <- Pedig$Born %in% (1980:1990) & Pedig$Breed=="Hinterwaelder"
Population <- Pedig$Indiv[use]

Pedig$NC   <- pedBreedComp(Pedig, thisBreed="Hinterwaelder")$native
pKin       <- pedIBD(Pedig, keep.only=Population)
pKinatN    <- pedIBDatN(Pedig, thisBreed="Hinterwaelder",  keep.only=Population)
Phen       <- Pedig[Population, ]

### Example 1: Overlapping Generations
### Old individuals contribute only little to the means:

cont <- agecont(Pedig, Population, maxAge=10)
cand <- candes(phen=Phen, pKin=pKin, pKinatN=pKinatN, cont=cont)

cand$current[,c("Name", "Type", "Breed", "Val", "Var")]
#     Name      Type         Breed         Val     Var
#1      BV     trait Hinterwaelder -0.55979308      BV
#2      NC     trait Hinterwaelder  0.56695077      NC
#3    pKin   kinship Hinterwaelder  0.02230896    pKin
#4 pKinatN nat. kin. Hinterwaelder  0.04678453 pKinatN

# BV:      simulated breeding values
# NC:      native genetic contribution computed from pedigree
# pKin:    pedigree-based kinship
# pKinatN: pedigree-based native kinship


### Example 2: Discrete Generations (cont=NULL).
### Old individuals and young individuals contribute equally to the means:

Phen$Born <- 1
cand <- candes(phen=Phen, pKin=pKin, pKinatN=pKinatN, cont=NULL)
```

```
cand$current[,c("Name", "Type", "Breed", "Val", "Var")]

#     Name      Type         Breed         Val     Var
#1      BV     trait Hinterwaelder -0.71910508      BV
#2      NC     trait Hinterwaelder  0.58226604      NC
#3    pKin   kinship Hinterwaelder  0.01979228    pKin
#4 pKinatN nat. kin. Hinterwaelder  0.04053012 pKinatN


### Shorthand:

cand$mean
#          BV       NC       pKin    pKinatN
#1 -0.7191051 0.582266 0.01979228 0.04053012

cand$mean$pKin
#[1] 0.01979228
```

`Cattle` *Phenotypes of Genotyped Cattle*

Description

Simulated phenotypes of cattle whose genotypes are included in files Chr1.phased, and Chr2.phased.

Usage

```
data(Cattle)
```

Format

Data frame containing information on genotyped cattle. The columns contain the ID of the individual (`Indiv`), the year of birth (`Born`), the breed name (`Breed`), a breeding value (`BV`), the sex (`Sex`), and the herd (`herd`).

`Chr1.phased` *Phased Cattle Genotypes from Chromosome 1*

Description

Phased genotypes of cattle from chromosome 1 (only the first part of the chromosome). Further information on these animals is included in data frame Cattle.

Format

All functions reading phased genotype data assume that the files are in the following format:

Each file has a header and no row names. Cells are separated by blank spaces. The number of rows is equal to the number of markers from the respective chromosome and the markers are in the same order as in the map. There can be some extra columns on the left hand side containing no genotype data. The remaining columns contain genotypes of individuals written as two alleles separated by a character, e.g. A/B, 0/1, A|B, A B, or 0 1. The same two symbols must be used for all markers. Column names are the IDs of the individuals. If the blank space is used as separator then the ID of each individual should be repeated in the header to get a regular delimited file. The columns to be skipped and the individual IDs must have no white spaces.

Use function read.indiv to extract the IDs of the individuals from a genotype file.

Examples

```
GTfile <- system.file("extdata/Chr1.phased", package="optiSel")
file.show(GTfile)
GT     <- read.table(GTfile, header=TRUE, skip=2, check.names=FALSE)
GT[1:10,1:5]
```

Chr2.phased *Phased Cattle Genotypes from Chromosome 2*

Description

Phased genotypes from Chromosome 2 (only the first part of the chromosome). Further information on these animals is included in data frame Cattle.

Format

All functions reading phased genotype data assume that the files are in the following format:

Each file has a header and no row names. Cells are separated by blank spaces. The number of rows is equal to the number of markers from the respective chromosome and the markers are in the same order as in the map. There can be some extra columns on the left hand side containing no genotype data. The remaining columns contain genotypes of individuals written as two alleles separated by a character, e.g. A/B, 0/1, A|B, A B, or 0 1. The same two symbols must be used for all markers. Column names are the IDs of the individuals. If the blank space is used as separator then the ID of each individual should be repeated in the header to get a regular delimited file. The columns to be skipped and the individual IDs must have no white spaces.

Use function read.indiv to extract the IDs of the individuals from a genotype file.

Examples

```
GTfile <- system.file("extdata/Chr2.phased", package="optiSel")
file.show(GTfile)
GT <- read.table(GTfile, header=TRUE, skip=2, check.names=FALSE)
GT[1:10,1:5]
```

`completeness` *Calculates Pedigree Completeness*

Description

Calculates completeness of the pedigree for individuals and for groups of individuals in each ancestral generation.

Usage

```
completeness(Pedig, keep=NULL, maxd=50, by="Indiv")
```

Arguments

`Pedig`	Data frame containing the pedigree, where the first columns are `Indiv` (Individual ID), `Sire`, and `Dam`. More columns can be passed in the `Pedig` argument, in particular a column for grouping with the name defined by argument `by`.
`keep`	Vector with IDs of the individuals for which the completeness will be calculated, or a logical vector indicating the individuals. By default, all individuals are used.
`maxd`	Number of generations for which completeness should be calculated.
`by`	Name of a column in data frame `Pedig`. The completeness will be computed separately for each group defined by the column.

Details

The function computes the completeness of the pedigree for the specified individuals and for groups of individuals. It is the proportion of known ancestors in each generation. Generation 0 corresponds to the individual itself, so the completeness is always 1 in generation 0.

Value

Data frame with the following columns

`Indiv (or 'by')`	ID of the individual or level of the grouping factor,
`Generation`	Generation number,
`Completeness`	Completeness of the pedigree in the respective generation.

Author(s)

Robin Wellmann

References

Cazes P, Cazes MH. (1996) Comment mesurer la profondeur genealogique d'une ascendance? Population (French Ed) 51:117-140.

See Also

Another function for characterizing pedigree completeness is summary.Pedig.

Examples

```
#Computes the pedigree completeness of Hinterwald cattle
#born between 2006 and 2007 in each ancestral generation.

data(PedigWithErrors)
Pedig <- prePed(PedigWithErrors)
compl <- completeness(Pedig, keep=Pedig$Born %in% (2006:2007), maxd=50, by="Indiv")
head(compl)

#Summary statistics can be computed directly from the pedigree:
Summary <- summary(Pedig, keep=Pedig$Born %in% (2006:2007))
head(Summary)

hist(Summary$PCI,        xlim=c(0,1),  main="Pedigree Completeness")
hist(Summary$Inbreeding, xlim=c(0,1),  main="Inbreeding")
hist(Summary$equiGen,    xlim=c(0,20), main="Number of Equivalent Complete Generations")
hist(Summary$fullGen,    xlim=c(0,20), main="Number of Fully Traced Generations")
hist(Summary$maxGen,     xlim=c(0,20), main="Number of Maximum Generations Traced")

compl <- completeness(Pedig, keep=Pedig$Born %in% (2006:2007), maxd=50, by="Sex")
head(compl)


library("ggplot2")
ggplot(compl, aes(Generation, Completeness, col=Sex))+geom_line()
```

---

conttac *Calculates **Cont**ributions **T**o **A**ge **C**ohorts*

Description

Calculates genetic contributions of other breeds to age cohorts

Usage

```
conttac(cont, cohort, use=rep(TRUE,length(cohort)), mincont=0.05, long=TRUE)
```

Arguments

cont	Data frame containing the genetic contributions of several ancestors or breeds to all individuals. This is typically the output of function pedBreedComp.
cohort	Numeric vector indicating for every individual the age cohort to which it belongs (typically year of birth).

use	Logical vector indicating for every individual whether it should be included in an age cohort (typically TRUE for individuals belonging to the breed of interest).
mincont	Contributions of breeeds with average contribution smaller than mincont will be summarized in one row
long	Should the resutling data frame be melted for easy plotting?

Details

The genetic contributions from other breeds to all age cohorts are computed. The genetic contribution from a breed is the fraction of genes in the gene pool originating from the respective breed.

Value

Data frame containing the genetic contribution from every breed to every age cohort.

Author(s)

Robin Wellmann

Examples

```
data(ExamplePed)
Pedig      <- prePed(ExamplePed, thisBreed="Hinterwaelder", lastNative=1970)
cont       <- pedBreedComp(Pedig, thisBreed="Hinterwaelder")
contByYear <- conttac(cont, Pedig$Born, use=Pedig$Breed=="Hinterwaelder", mincont=0.04, long=FALSE)
round(contByYear,2)

barplot(contByYear, ylim=c(0,1), col=1:10, ylab="genetic contribution",
        legend=TRUE, args.legend=list(x="bottomleft",cex=0.5))
```

ExamplePed *Pedigree of Hinterwald Cattle*

Description

This data set gives a small subset of the pedigree of Hinterwald cattle suitable for demonstration purposes.

Usage

```
data(ExamplePed)
```

Format

Data frame with columns Indiv (individual ID), Sire, Dam, Sex, Breed, Born with year of birth, and simulated breeding value BV.

freqlist *Combines Objects Computed with Function haplofreq() into a List*

Description

The function combines objects computed with function haplofreq into a list with class HaploFreq and adds some attributes.

Usage

```
freqlist(...)
```

Arguments

...	R-objects computed with function haplofreq.

Details

The function combines objects computed with function haplofreq into a list with class HaploFreq.

Value

A list with class HaploFreq

Author(s)

Robin Wellmann

Examples

```
data(map)
data(Cattle)
dir   <- system.file("extdata", package="optiSel")
files <- paste(dir, "/Chr", 1:2, ".phased", sep="")

Freq <- freqlist(
 haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="Rotbunt",   minL=2.0),
 haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="Holstein",  minL=2.0),
 haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="Fleckvieh", minL=2.0)
  )

#The component names are the reference breeds by default:
names(Freq)

plot(Freq, ID=1, hap=2, refBreed="Rotbunt")

plot(Freq, ID=1, hap=2, refBreed="Holstein", Chr=1)
```

genecont *Calculates Genetic Contributions using Pedigrees.*

Description

Calculates the genetic contributions each individual has from specified ancestors.

Usage

```
genecont(Pedig, from=NULL, to=NULL)
```

Arguments

Pedig	Data frame containing the pedigree, where the first columns are Indiv (Individual ID), Sire, and Dam.
from	Vector with ancestors whose contributions to the individuals should be calculated. By default, the contributions from all individuals will be calculated.
to	Vector with individuals for which the contributions from ancestors should be calculated. By default, the contributions are calculated for all individuals.

Details

This function calculates genetic contributions of specified ancestors to each individual.

Value

Lower triangular matrix with genetic contributions for each pair of individuals. Column i contains the genetic contribution of ancestor i to all individuals.

Author(s)

Robin Wellmann

Examples

```
data(ExamplePed)
Pedig <- prePed(ExamplePed)
cont  <- genecont(Pedig)

plot(Pedig$Born, cont[,"276000803611144"], pch=18, ylim=c(0,1))
Pedig["276000803611144",]

#faster:
cont  <- genecont(Pedig, from="276000803611144")
head(cont)
plot(Pedig$Born, cont[,"276000803611144"], pch=18, ylim=c(0,1))
```

haplofreq *Evaluates the Occurrence of Haplotype Segments in Particular Breeds*

Description

For each haplotype from thisBreed and every SNP the occurence of the haplotype segment containing the SNP in a set of reference breeds is evaluated. The maximum frequency each segment has in one of these reference breeds is computed, and the breed in which the segment has maximum frequency is identified. Results are either returned in a list or saved to files.

Usage

```
haplofreq(files, phen, map, thisBreed, refBreeds="others", minSNP=20, minL=1.0,
  unitL="Mb", ubFreq=0.01, keep=NULL, skip=NA, cskip=NA, w.dir=NA,
  what=c("freq", "match"), cores=1, quiet=FALSE)
```

Arguments

files	Either a character vector with file names, or a list containing character vectors with file names. The files contain phased genotypes, one file for each chromosome. File names must contain the chromosome name as specified in the map in the form "ChrNAME.", e.g. "Breed2.Chr1.phased". The required format of the marker files is described under Details. If file is a character vector then, genotypes of all animals must be in the same files. Alternatively, files can be a list with the following two components: hap.thisBreed: Character vector with names of the phased marker files for the individuals from thisBreed, one file for each chromosome. hap.refBreeds: Character vector with names of the phased marker files for the individuals from the reference breeds (refBreeds), one file for each chromosome. If this component is missing, then it is assumed that the haplotypes of these animals are also included in hap.thisBreed.
phen	Data frame containing the ID (column "Indiv") and the breed name (column "Breed") of each genotyped individual.
map	Data frame providing the marker map with columns including marker name 'Name', chromosome number 'Chr', and possibly the position on the chromosome in mega base pairs 'Mb', and the position in centimorgan 'cM'. The order of the markers must be the same as in the files files. Marker names must have no white spaces.
thisBreed	Name of a breed from column Breed in phen: The occurence of each haplotype segment from this breed in the reference breeds will be evaluated.
refBreeds	Vector with names of breeds from column Breed in phen. These breeds are used as reference breeds. The occurence of haplotype segments in these breeds will be evaluated. By default, all breeds in phen, except thisBreed are used as reference breeds. In contrast, for refBreeds="all", all genotyped breeds are used as reference breeds.

minSNP	Minimum number of marker SNPs included in a segment.
minL	Minimum length of a segment in unitL (e.g. in cM or Mb).
unitL	The unit for measuring the length of a segment. Possible units are the number of marker SNPs included in the segment ('SNP'), the number of mega base pairs ('Mb'), and the genetic distances between the first and the last marker in centiMorgan ('cM'). In the last two cases the map must include columns with the respective names.
ubFreq	If a haplotype segment has frequency smaller than ubFreq in all reference breeds then the breed name is replaced by '1', which indicates that the segment is native.
keep	Subset of the IDs of the individuals from data frame phen, or a logical vector indicating the animals in data frame phen that should be used. The default keep=NULL means that all individuals included in phen will be considered.
skip	Take line skip+1 of the files as the line with column names. By default, the number is determined automatically.
cskip	Take column cskip+1 of the files as the first column with genotypes. By default, the number is determined automatically.
w.dir	Output file directory. Writing results to files has the advantage that much less working memory is required. By default, no files are created. The function returns only the file names if files are created.
what	For what="freq", the maximum frequency each haplotype segment has in the reference breeds will be computed. For what="match", the name of the reference breed in which the segment has maximum frequency will be determined. By default, the frequencies and the breed names both are determined.
cores	Number of cores to be used for parallel processing of chromosomes. By default one core is used. For cores=NA the number of cores will be chosen automatically. Using more than one core increases execution time if the function is already fast.
quiet	Should console output be suppressed?

Details

For each haplotype from thisBreed and every SNP the occurence of the haplotype segment containing the SNP in a set of reference breeds is evaluated. The maximum frequency each segment has in one of these reference breeds is computed, and the breed in which the segment has maximum frequency is identified. Results are either returned in a list or saved to files.

Marker file format: Each marker file containing phased genotypes has a header and no row names. Cells are separated by blank spaces. The number of rows is equal to the number of markers from the respective chromosome and the markers are in the same order as in the map. The first cskip columns are ignored. The remaining columns contain genotypes of individuals written as two alleles separated by a character, e.g. A/B, 0/1, A|B, A B, or 0 1. The same two symbols must be used for all markers. Column names are the IDs of the individuals. If the blank space is used as separator then the ID of each individual should repeated in the header to get a regular delimited file. The columns to be skipped and the individual IDs must have no white spaces.

Value

If w.dir=NA then a list is returned. The list may have the following components:

freq	Mx(2N) - matrix containing for every SNP and for each of the 2N haplotypes from thisBreed the maximum frequency the segment containing the SNP has in a the reference breeds.
match	Mx(2N) - matrix containing for every SNP and for each of the 2N haplotypes from thisBreed the first letter of the name of the reference breed in which the segment containing the SNP has maximum frequency. Segments with frequencies smaller than ubFreq in all reference breeds are marked as '1', which indicates that the segment is native for thisBreed.

The list has attributes thisBreed, and map.

If w.dir is the name of a directory, then results are written to files, whereby each file corresponds to one chromosome, and a data frame with file names is returned.

Author(s)

Robin Wellmann

Examples

```
data(map)
data(Cattle)
dir   <- system.file("extdata", package="optiSel")
files <- file.path(dir, paste("Chr", 1:2, ".phased", sep=""))

Freq <- freqlist(
 haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="Rotbunt",   minL=2.0),
 haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="Holstein",  minL=2.0),
 haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="Fleckvieh", minL=2.0)
  )

plot(Freq, ID=1, hap=2, refBreed="Rotbunt")
plot(Freq, ID=1, hap=2, refBreed="Holstein", Chr=1)


## Test for using multiple cores:

Freq1 <- haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="Rotbunt",
                   minL=2.0, cores=NA)$freq
range(Freq[[1]]-Freq1)
#[1] 0 0


## Creating output files with allele frequencies and allele origins:

rdir  <- system.file("extdata", package = "optiSel")
wdir  <- file.path(tempdir(), "HaplotypeEval")
chr   <- unique(map$Chr)
files <- file.path(rdir, paste("Chr", chr, ".phased", sep=""))
```

```
wfile <- haplofreq(files, Cattle, map, thisBreed="Angler", minL=2.0, w.dir=wdir)

View(read.table(wfile$match[1],skip=1))
#unlink(wdir, recursive = TRUE)
```

---

makeA *Calculates the Pedigree-based Additive Relationship Matrix*

Description

Calculates the the Pedigree-based Additive Relationship Matrix. This is twice the pedigree based kinship matrix.

Usage

```
makeA(Pedig, keep.only=NULL, keep=keep.only, AFounder=NULL)
```

Arguments

Pedig	Data frame containing the Pedigree. The data frame has columns (1) Individual, (2) Sire, (3) Dam. Missing parents are coded as NA. Both parents must either be missing or present. If this is not the case use prePed.
keep	If keep is provided then kinships are computed only for these animals and their ancestors.
keep.only	If keep.only is provided then kinships are computed only for these animals.
AFounder	Additive relationship matrix of the founders. The row names are the ids of the founders. By default, founders are assumed to be unrelated. Founders not included in this matrix are also assumed to be unrelated.

Details

Computation of pedigree based additive relationship matrix A which is twice the kinship matrix. For individuals i and j it is defined as

Aij = 2*(Probability that two alleles chosen from individuals i and j are IBD).

Value

Additive relationship matrix.

Author(s)

Robin Wellmann

Examples

```
data(PedigWithErrors)
data(Phen)
Pedig <- prePed(PedigWithErrors)
keep  <- Pedig$Indiv[summary(Pedig)$equiGen>5 & Pedig$Indiv %in% Phen$Indiv]
A     <- makeA(Pedig, keep.only=keep)
A[1:3,1:3]
```

---

map *Marker Map for Cattle*

Description

Marker map for SNPs from cattle chromosomes 1 - 2 (only the first parts of the chromosomes). The corresponding genotypes are included in Chr1.phased and Chr2.phased.

Usage

```
data(map)
```

Format

Data frame containing the marker map including marker name (`Name`), chromosome number (`Chr`), position in base pairs (`Position`), position in centiMorgan (`cM`), and position in mega base pairs (`Mb`).

---

matings *Mate Allocation*

Description

Males and females are allocated for mating such that all breeding animals have the desired number of matings. The mean inbreeding coefficient in the offspring is minimized if matrix `Kin` contains pairwise kinships of the selection candidates.

Usage

```
matings(phen, Kin,  alpha=1,
    ub.n=NA, max=FALSE, solver="default", ...)
```

Arguments

phen	Data frame with desired number of matings (column n), sexes (column Sex), and IDs (column Indiv) of the selection candidates. The data frame may also contain column herd containing the names of the herds to which the females belong (NA for males).
Kin	Kinship matrix (or an other similarity matrix) for selection candidates.
alpha	If alpha<1 then the proportion of matings with the same male is at most alpha in each herd. A value alpha<1 increases genetic connectedness between herds and enables to estimate more accurate breeding values.
ub.n	Maximum number of matings of the same individuals. Without this constraint (i.e. ub.n=NA), some superior animals may always be mated to the same inferior animal, so their offspring would likely not be suitable for breeding.
max	The default max=FALSE means that the objective function is minimized.
solver	Either solver="default", or solver=Rsymphony_solve_LP. The latter is possible only if package Rsymbhony is loaded, which is not available for all platforms.
...	Further optimization parameters. By default, they are passed to function ecos.control
.	

Details

Males and females are allocated for mating such that all breeding animals have the desired number of matings. If Kin is a kinship matrix, then the mean inbreeding coefficient in the offspring is minimized. In general, the mean similarity of the parents is minimized.

The maximum number of matings of the same individuals can be constrained. For each herd, the proportion alpha of matings with the same male can be constrained as well, but this increases computation time.

Value

Data frame with columns Sire, Dam, n, and possibly herd, whereby column n contains the desired number of matings, and column herd contains the herd of the dam.

The data frame has attributes objval with the value of the objective function (usually the mean inbreeding coefficient), and attribute info describing the solution as reported by the solver.

Author(s)

Robin Wellmann

Examples

```
data("map")
data("Cattle")
dir   <- system.file("extdata", package = "optiSel")
files <- paste(dir, "/Chr", 1:2, ".phased", sep="")
```

```
sKin  <- segIBD(files, map, minSNP=20, minL=2.0)
Phen  <- Cattle[Cattle$Breed=="Angler", ]

cont  <- data.frame(
  age   = c(   1,    2,    3,    4,    5,    6),
  male  = c(0.11, 0.11, 0.10, 0.08, 0.06, 0.04),
  female= c(0.11, 0.11, 0.10, 0.08, 0.06, 0.04))

cand  <- candes(phen=Phen, sKin = sKin, cont=cont)
con   <- list(uniform="female", ub.sKin = 0.047)
Offspring <- opticont("max.BV", cand, con, trace=FALSE)


#####  Minimize inbreeding   #####
Candidate   <- Offspring$parent
Candidate$n <- noffspring(Candidate, N=20)$nOff
Mating      <- matings(Candidate, sKin)
Mating
attributes(Mating)$objval



library("Rsymphony")
Mating <- matings(Candidate, sKin, alpha=0.30, solver=Rsymphony_solve_LP)
Mating
attributes(Mating)$objval

attributes(Mating)$info
#[1] "Optimum solution found"
```

`noffspring` *Calculates Optimum Numbers of Offspring*

Description

Calculates the optimum numbers of offspring from optimum contributions of selection candidates.

Usage

```
noffspring(cand, N, random=TRUE)
```

Arguments

`cand`	Data frame with optimum contributions (column `oc`), sexes (column `Sex`), and IDs (column `Indiv`) of the selection candidates.
`N`	Desired number of individuals in the offspring population.
`random`	Logical. If `2*N*oc[i]` is not an integer value (say `2*N*oc[i]=11.4`) then individual `i` will have either 11 or 12 offspring. The actual number is either determined randomly or not.

Details

The function calculates the optimum numbers of offspring of the selection candidates from the optimum contributions cand$oc and the size N of the offspring population.

Value

Data frame with column Indiv containing the individual IDs and column nOff containing the optimum numbers of offspring.

Column nOff is approximately 2*N*cand$oc with sum(noff[cand$Sex=="male"])=N and sum(noff[cand$Sex=="female"

Author(s)

Robin Wellmann

Examples

```
set.seed(1)
data(PedigWithErrors)

Pedig      <- prePed(PedigWithErrors, thisBreed="Hinterwaelder")
use        <- Pedig$Born %in% (1998:2008) & Pedig$Breed=="Hinterwaelder"
Population <- sampleIndiv(Pedig[use, ], each=50)
pKin       <- pedIBD(Pedig, keep.only=Population)
Phen       <- Pedig[Population, ]
Phen$isCandidate <- Phen$Born %in% (2003:2008)

cont       <- agecont(Pedig, Population)
cand       <- candes(phen=Phen, fA=pedIBD(Pedig, keep.only=Phen$Indiv), cont=cont)
con        <- list(ub.fA=0.0175, uniform="female")
Offspring  <- opticont("max.BV", cand, con, trace = FALSE)

N <- 250
Candidate <- Offspring$parent
Candidate$nOff <- noffspring(Candidate, N)$nOff

sum(Candidate$nOff[Candidate$Sex=="male"])
#[1] 250

sum(Candidate$nOff[Candidate$Sex=="female"])
#[1] 250

round(2*N*Candidate$oc-Candidate$nOff, 2)
```

---

opticomp *Calculates the Optimum Breed Composition*

Description

Calculates optimum contributions of breeds to a hypothetical multi-breed population with maximum diversity. Additionally the average kinship within and between breeds and the genetic distances between breeds are computed.

Usage

```
opticomp(f, phen, obj.fun="NGD", lb=NULL, ub=NULL, ...)
```

Arguments

`f`	Kinship matrix (e.g. a segment based kinship matrix).
`phen`	Data frame with column `Indiv` containing the IDs of the individuals and `Breed` with breed names.
`obj.fun`	The objective function to be maximized. For `"NGD"` the objective is to maximize the genetic diversity 1-**c**'**fc** in the multi-breed population, where **f** is the matrix containing the mean kinships within and between breeds. For `"NTD"` the term **c**'(**1**-**F**)+**c**'(**F1**' - 2**f** + **1F**')**c** is maximized, where **F**=diag(**f**). This puts more weight on between population diversity.
`lb`	Named vector providing lower bounds for the contributions of the breeds can be provided. The names of the components are the breed names. The default `lb=NULL` means that the lower bound is 0 for all breeds.
`ub`	Named vector providing upper bounds for the contributions of the breeds can be provided. The names of the components are the breed names. The default `ub=NULL` means that the upper bound is 1 for all breeds.
`...`	Further parameters passed to the solver solve.QP of R package `quadprog`.

Details

Calculates optimum contributions of breeds to a hypothetical multi-breed population with maximum diversity. Additionally the average kinship within and between breeds and the genetic distances between breeds are computed.

Value

A list with the following components:

`bc`	Vector with optimum contributions of breeds to a hypothetical multi-breed population with maximum diversity
`value`	The value of the objective function, i.e. the maximum diversity that can be achieved.
`f`	Matrix containing the mean kinships within and between breeds.
`Dist`	Genetic distances between breeds.

Author(s)

Robin Wellmann

References

Wellmann, R., Bennewitz, J., Meuwissen, T.H.E. (2014) A unified approach to characterize and conserve adaptive and neutral genetic diversity in subdivided populations. Genetics Selection Evolution. 69, e16

Examples

```
library(optiSel)
data(map)
data(Cattle)
dir   <- system.file("extdata", package = "optiSel")
files <- paste(dir, "/Chr", 1:2, ".phased", sep="")

########################################################################
#   Find the optimum breed composition using segment based kinship  #
########################################################################
IBD <- segIBD(files, minSNP=20, map=map, minL=2.0)
mb  <- opticomp(IBD, Cattle, obj.fun="NGD")

#### Optimum breed composition: ###
round(mb$bc,3)
#   Angler Fleckvieh  Holstein   Rotbunt
#    0.469     0.444     0.041     0.046

#### Average kinships within and between breeds: ###
round(mb$f,4)
#          Angler Fleckvieh Holstein Rotbunt
#Angler    0.0523    0.0032   0.0414  0.0417
#Fleckvieh 0.0032    0.0625   0.0036  0.0032
#Holstein  0.0414    0.0036   0.1074  0.0894
#Rotbunt   0.0417    0.0032   0.0894  0.1057

#### Genetic distances between breeds: ###
round(mb$Dist,4)
#         Angler Fleckvieh Holstein Rotbunt
#Angler    0.0000    0.2329   0.1960  0.1930
#Fleckvieh 0.2329    0.0000   0.2853  0.2844
#Holstein  0.1960    0.2853   0.0000  0.1309
#Rotbunt   0.1930    0.2844   0.1309  0.0000

########################################################################
#   The optimum breed composition depends on the kinship matrix     #
#   and the objective function:                                     #
########################################################################

bc <- opticomp(IBD, Cattle, obj.fun="NTD")$bc
round(bc,3)
#   Angler Fleckvieh  Holstein   Rotbunt
#    0.264     0.447     0.148     0.141
```

opticont *Optimum Contributions of Selection Candidates*

Description

The optimum contributions of selection candidates to the offspring are calculated. The optimization procedure can take into account conflicting breeding goals, which are to achieve genetic gain, to reduce the rate of inbreeding, and to recover the original genetic background of a breed.

It can be used for overlapping as well as for non-overlapping generations. In the case of overlapping generations, average values of the parameters for the population in the next year will be optimized, whereas for non-overlapping generations, average values of the parameters in the next generation will be optimized. Below, the "next evaluation time" means the next year for populations with overlapping generations, but the next generation for populations with non-overlapping generations.

Optimization can be done for several breeds or breeding lines simultaneously, which is adviseable if the aim is to increase diversity or genetic distance between them.

Usage

```
opticont(method, cand, con, bc=NULL,  solver="default", quiet=FALSE,
         make.definite=FALSE, ...)
```

Arguments

method	Character string "min.VAR", or "max.VAR", whereby VAR is the name of the variable to be minimized or maximized. Available methods are reported by function candes.
cand	An R-Object containing all information describing the individuals (phenotypes and kinships). These indivdiuals are a sample from the population that includes the selection candidates. It can be created with function candes. This object also defines whether generations are overlapping or non-overlapping. * If the aim is to increase genetic distance between breeds, then samples from several breeds are needed. * If column Sex of data frame cand$phen contains NA for one breed, then the constraint stating that contributions of both sexes must be equal is omitted.
con	List defining threshold values for constraints. The components are described in the Details section. If one is missing, then the respective constraint is not applied. Permitted constraint names are reported by function candes.
bc	Named numeric vector with breed contributions, which is only needed if cand$phen contains individuals from different breeds. It contains the proportion of each breed in a hypothetical multi-breed population for which the diversity across breeds should be managed. The names of the components are the breed names.
solver	Name of the solver used for optimization. Available solvers are "alabama", "cccp", "cccp2", and "slsqp". Solver "csdp" is disabled because the package Rcsdp has been removed from Cran. By default, the solver is chosen automatically. The solvers are the same as for function solvecop from package optiSolve.

quiet	If quiet=FALSE then detailed information is shown.
make.definite	Logical variable indicating whether non-positive-semidefinite matrices should be approximated by positive-definite matrices. This is always done for solvers that are known not to convergue otherwise.
...	Tuning parameters of the solver. The available parameters depend on the solver and will be printed when function opticont is used with default values. Definitions of the tuning parameters can be found for alabama in auglag and optim, for cccp and cccp2 in ctrl, and for slsqp in nl.opts.

Details

The optimum contributions of selection candidates to the offspring are calculated. The proportion of offspring that should have a particular selection candidate as parent is twice its optimum contribution.

Constraints

Argument con is a list defining the constraints. Permitted names for the components are displayed by function candes. Their meaning is as follows:

uniform: Character vector specifying the breeds or sexes for which the contributions are not to be optimized. Within each of these groups it is assumed that all individuals have equal (uniform) contributions. Character string "BREED.female" means that all females from breed BREED have equal contributions and thus equal numbers of offspring. Column 'isCandidate' of cand$phen is ignored for these individuals.

lb: Named numeric vector containing lower bounds for the contributions of the selection candidates. The component names are their IDs. By default the lower bound is 0 for all individuals.

ub: Named numeric vector containing upper bounds for the contributions of the selection candidates. Their component names are the IDs. By default no upper bound is specified.

ub.VAR: Upper bound for the expected mean value of kinship or trait **VAR** in the population at the next evaluation time. Upper bounds for an arbitrary number of different kinships and traits may be provided. If data frame cand$phen contains individuals from several breeds, the bound refers to the mean value of the kinship or trait in the multi-breed population.

ub.VAR.BREED: Upper bound for the expected mean value of kinship or trait **VAR** in the breed **BREED** at the next evaluation time. Upper bounds for an arbitrary number of different kinships and traits may be provided.

Note that **VAR** must be replaced by the name of the variable and **BREED** by the name of the breed. For traits, lower bounds can be defined as **lb.VAR** or **lb.VAR.BREED**. Equality constraints can be defined as **eq.VAR** or **eq.VAR.BREED**.

Application to multi-breed data

Optimization can be done for several breeds or breeding lines simultaneously, which is adviseable if the aim is to increase genetic diversity in a multi-breed population, or to increase the genetic distances between breeds or breeding lines. However, for computing the kinship of individuals from different breeds, marker data is needed.

The multi-breed population referred above is a hypothetical subdivided population consisting of purebred animals from the breeds included in column Breed of cand$phen. The proportion of individuals from a given breed in this population is its breed contribution specified in argument bc. It is not the proportion of individuals of this breed in data frame cand$phen.

The aim is to minimize or to constrain the average genomic kinship in this multi-breed population. This causes the genetic distance between the breeds to increase, and thus may increase the conservation value of the breeds, or the heterosis effects in crossbred animals.

Remark

If the function does not provide a valid result due to numerical problems then try to use another solver, use other optimization parameters, define upper or lower bounds instead of equality constraints, or relax the constraints to ensure that the optimization problem is solvable.

Value

A list with the following components

`parent`	Data frame `cand$phen` with some appended columns. Column `oc` contains the optimum contributions of the selection candidates, column `lb` the lower bounds, and `ub` the upper bounds for the contributions.
`info`	Data frame with component `valid` indicating if all constraints are fulfilled, component `solver` containing the name of the solver used for optimization, and component `status` describing the solution as reported by the solver.
`mean`	Data frame containing the expected mean value of each kinship and trait in the population at the next evaluation time.
`bc`	Data frame with breed contributions in the hypothetical multi-breed population used for computing the average kinship across breeds.
`obj.fun`	Named numeric value with value and name of the objective function.
`summary`	Data frame containing one row for each constraint with the value of the constraint in column `Val`, and the bound for the constraint in column `Bound`. Column `OK` states if the constraint is fulfilled, and column `Breed` contains the name of the breed to which the constraint applies. The value of the objective function is shown in the first row. Additional rows contain the mean values of traits and kinships in the population at the next evaluation time which are not constrained.

Author(s)

Robin Wellmann

References

Wellmann, R. (2018). Optimum Contribution Selection and Mate Allocation for Breeding: The R Package optiSel. submitted

Examples

```
## For other objective functions and constraints see the vignettes


######################################################
# Example 1: Advanced OCS with overlapping           #
#            generations using pedigree data         #
#   - maximize genetic gain                (BV)     #
```

```
#   - restrict increase of mean kinship     (pKin)   #
#   - restrict increase of native kinship   (pKinatN)#
#   - avoid decrease of native contribution (NC)     #
#####################################################

### Define object cand containing all required
### information on the individuals

data(PedigWithErrors)
Pedig    <- prePed(PedigWithErrors, thisBreed="Hinterwaelder", lastNative=1970,
                   keep=PedigWithErrors$Born%in%1992)
Pedig$NC <- pedBreedComp(Pedig, thisBreed="Hinterwaelder")$native
use      <- Pedig$Born %in% (1980:1990) & Pedig$Breed=="Hinterwaelder"
use      <- use & summary(Pedig)$equiGen>=3
cont     <- agecont(Pedig, use, maxAge=10)

Phen     <- Pedig[use, ]
pKin     <- pedIBD(Pedig, keep.only=Phen$Indiv)
pKinatN  <- pedIBDatN(Pedig, thisBreed="Hinterwaelder",  keep.only=Phen$Indiv)
Phen$isCandidate <-  Phen$Born < 1990
cand     <- candes(phen=Phen, pKin=pKin, pKinatN=pKinatN, cont=cont)

### Mean values of the parameters in the population:

cand$mean
#          BV        NC      pKin    pKinatN
#1 -0.5648208 0.5763161 0.02305245 0.0469267


### Define constraints for OCS
### Ne: Effective population size
### L:  Generation interval

Ne   <- 100
L    <- 1/(4*cont$male[1]) + 1/(4*cont$female[1])
con <- list(uniform    = "female",
            ub.pKin    = 1-(1-cand$mean$pKin)*(1-1/(2*Ne))^(1/L),
            ub.pKinatN = 1-(1-cand$mean$pKinatN)*(1-1/(2*Ne))^(1/L),
            lb.NC      = cand$mean$NC)

### Solve the optimization problem

Offspring  <- opticont("max.BV", cand, con, trace=FALSE)

### Expected average values of traits and kinships
### in the population now and at the next evaluation time

rbind(cand$mean, Offspring$mean)
#          BV        NC       pKin    pKinatN
#1 -0.5648208 0.5763161 0.02305245 0.04692670
#2 -0.4972679 0.5763177 0.02342014 0.04790944

### Data frame with optimum contributions
```

```
Candidate <- Offspring$parent
Candidate[Candidate$oc>0.01, c("Indiv", "Sex", "BV", "NC", "lb", "oc", "ub")]


#####################################################
# Example 2: Advanced OCS with overlapping          #
#            generations using genotype data        #
#   - minimize mean kinship                (sKin)   #
#   - restrict increase of native kinship  (sKinatN)#
#   - avoid decrease of breeding values    (BV)     #
#   - cause increase of native contribution (NC)    #
#####################################################


### Prepare genotype data

data(map)
data(Cattle)

### Compute genomic kinship and genomic kinship at native segments
dir     <- system.file("extdata", package = "optiSel")
files   <- file.path(dir, paste("Chr", 1:2, ".phased", sep=""))
sKin    <- segIBD(files, map, minL=1.0)
sKinatN <- segIBDatN(files, Cattle, map, thisBreed="Angler",  minL=1.0)

### Compute migrant contributions of selection candidates
Haplo   <- haplofreq(files, Cattle, map, thisBreed="Angler", minL=1.0, what="match")
Comp    <- segBreedComp(Haplo$match, map)
Cattle[Comp$Indiv, "NC"] <- Comp$native

Phen  <- Cattle[Cattle$Breed=="Angler",]
cand  <- candes(phen=Phen, sKin=sKin, sKinatN=sKinatN, cont=cont)

### Define constraints for OCS
### Ne: Effective population size
### L:  Generation interval

Ne <- 100
L  <- 4.7
con <- list(uniform    = "female",
            ub.sKinatN = 1-(1-cand$mean$sKinatN)*(1-1/(2*Ne))^(1/L),
            lb.NC      = 1.03*cand$mean$NC,
            lb.BV      = cand$mean$BV)

# Compute optimum contributions; the objective is to minimize mean kinship
Offspring   <- opticont("min.sKin", cand, con=con)

# Check if the optimization problem is solved
Offspring$info

# Average values of traits and kinships
rbind(cand$mean, Offspring$mean)
```

```
#           BV        NC       sKin    sKinatN
#1 -0.07658022 0.4117947 0.05506277 0.07783431
#2 -0.07657951 0.4308061 0.04830328 0.06395410

# Value of the objective function
Offspring$obj.fun
#      sKin
#0.04830328

### Data frame with optimum contributions

Candidate <- Offspring$parent
Candidate[Candidate$oc>0.01, c("Indiv", "Sex", "BV", "NC", "lb", "oc", "ub")]


#########################################################
# Example 3: Advanced OCS with overlapping              #
#            generations using genotype data            #
#            for multiple breeds or beeding lines       #
#   - Maximize breeding values in all breeds            #
#   - restrict increase of kinships within each breed   #
#   - reduce average kinship across breeds              #
#   - restrict increase of native kinship in Angler     #
#   - cause increase of native contribution in Angler   #
# by optimizing contributions of males from all breeds  #
#########################################################


cand <- candes(phen=Cattle, sKin=sKin, sKinatN.Angler=sKinatN, cont=cont)
L  <- 5
Ne <- 100

con  <- list(uniform          = "female",
             ub.sKin          = cand$mean$sKin - 0.01/L,
             ub.sKin.Angler   = 1-(1-cand$mean$sKin.Angler)*(1-1/(2*Ne))^(1/L),
             ub.sKin.Holstein = 1-(1-cand$mean$sKin.Holstein)*(1-1/(2*Ne))^(1/L),
             ub.sKin.Rotbunt  = 1-(1-cand$mean$sKin.Rotbunt)*(1-1/(2*Ne))^(1/L),
             ub.sKin.Fleckvieh= 1-(1-cand$mean$sKin.Fleckvieh)*(1-1/(2*Ne))^(1/L),
             ub.sKinatN.Angler= 1-(1-cand$mean$sKinatN.Angler)*(1-1/(2*Ne))^(1/L),
             lb.NC            = cand$mean$NC + 0.05/L)

Offspring <- opticont("max.BV", cand, con, trace=FALSE, solver="slsqp")

Offspring$mean


Candidate <- Offspring$parent[Offspring$parent$Sex=="male", ]
Candidate[Candidate$oc>0.01, c("Indiv", "Sex", "BV", "NC", "lb", "oc", "ub")]
```

pedBreedComp *Calculates the Pedigree Based Breed Composition of Individuals*

Description

Computes for every individual the genetic contribution from native founders and from other breeds according to the pedigree.

Usage

```
pedBreedComp(Pedig, thisBreed)
```

Arguments

Pedig	Data frame containing the pedigree with the first 3 columns being Indiv (individual ID), Sire, and Dam. Additional columns include column Breed with breed names. Missing parents are coded as NA. All animals have no parent or both parents missing. It is usually created with function prePed.
thisBreed	Name of the breed of interest as denoted in column Breed of the pedigree.

Details

For every individual the genetic contribution from native founders and from other breeds is computed. It is the fraction of genes that originate from the respective breed.

Value

Data frame with one row for each individual and the following columns

Indiv	IDs of the individuals
native	Native Contribution: The genetic contribution from native founders.
...	Genetic contributions from other breeds, one column for each breed. The columns are ordered, so that the most influential breeds come first.

Author(s)

Robin Wellmann

Examples

```
data(ExamplePed)
Pedig    <- prePed(ExamplePed, thisBreed="Hinterwaelder", lastNative=1970)
cont     <- pedBreedComp(Pedig, thisBreed="Hinterwaelder")
cont[1000:1010,2:5]

contByYear <- conttac(cont, Pedig$Born, use=Pedig$Breed=="Hinterwaelder", mincont=0.04, long=FALSE)
round(contByYear,2)
```

```
barplot(contByYear,ylim=c(0,1), col=1:10, ylab="genetic contribution",
        legend=TRUE, args.legend=list(x="bottomleft",cex=0.6))
```

---

pedIBD *Calculates the Pedigree-based Kinship Matrix*

Description

Calculates the **ped**igree based probability of alleles to be **IBD**. This pedigree based kinship matrix is also called coancestry matrix and is half the additive relationship matrix.

Usage

```
pedIBD(Pedig, keep.only=NULL, keep=keep.only, kinFounder=NULL)
```

Arguments

`Pedig`	Data frame containing the pedigree with `Indiv` (individual ID), `Sire`, and `Dam` in the first 3 columns. Missing parents are coded as NA. Both parents must either be missing or present. If this is not the case use function prePed to prepare the pedigree.
`keep`	If `keep` is provided then kinships are computed only for these animals and their ancestors.
`keep.only`	If `keep.only` is provided then kinships are computed only for these animals.
`kinFounder`	Kinship matrix for the founders. The row names are the ids of the founders. By default, founders are assumed to be unrelated. Founders not included in this matrix are also assumed to be unrelated.

Details

Computation of pedigree based kinship matrix f which is half the additive relationship matrix. For individuals i and j it is defined as

fij = Probability that two alleles chosen from individuals i and j are IBD.

Value

Kinship matrix.

Author(s)

Robin Wellmann

Examples

```
data(PedigWithErrors)
data(Phen)
keep  <- Phen$Indiv
Pedig <- prePed(PedigWithErrors, keep=keep, thisBreed="Hinterwaelder", lastNative=1970)
pedA  <- pedIBD(Pedig, keep.only=keep)
```

---

pedIBDatN *Calculates the Pedigree Based Kinship at Native Alleles*

Description

Calculates the kinship at native alleles, which is the pedigree based probability of native alleles to be IBD.

Usage

```
pedIBDatN(Pedig, thisBreed=NA, keep.only=NULL, keep=keep.only, nGen=NA, quiet=FALSE)
```

Arguments

Pedig	Data frame containing the pedigree with Indiv (Individual ID), Sire, and Dam in the first 3 columns, column Breed with breed names, and possibly column Sex. Missing parents are coded as NA, 0, or "0".
thisBreed	Name of the breed for which the kinships are to be computed.
keep	If keep is provided then kinships are computed only for these animals and their ancestors.
keep.only	If keep.only is provided then kinships are computed only for these animals.
nGen	Number of generations taken into account for estimating the native effective size. The default means that the native effective size is not estimated, which requires less memory.
quiet	Should console output be suppressed?

Details

Calculates a list containing matrices needed to compute pedigree based kinships at native alleles, defined as the conditional probability that two randomly chosen alleles are IBD, given that both originate from native founders. A native founder is an individual with unkown parents belonging to thisBreed.

The kinship at native alleles between individuals i and j is Q1[i,j]/Q2[i,j].

The mean kinship at native alleles in the offspring is (x'Q1x+d1)/(x'Q2x+d2), where x is the vector with genetic contributions of the selection candidates.

The native effective size is estimated from nGen generations only if nGen is not NA.

Value

A list of class `ratioFun` including components:

`Q1`	matrix with `Q1[i,j]` = Probability that two alleles chosen from individuals i and j are IBD and are native.
`Q2`	matrix with `Q2[i,j]` = Probability that two alleles chosen from individuals i and j are both native.
`d1`	The value by which the probability that two alleles chosen from the offspring are IBD and native increases due to genetic drift.
`d2`	The value by which the probability that two alleles chosen from the offspring are native increases due to genetic drift.
`id`	IDs of the individuals for which the probabilites have been computed.
`mean`	Mean kinship at native alleles of the individuals specified in argument `keep.only`. Note that `1-mean` is the genetic diversity at native segments of the specified individuals from `thisBreed`

Author(s)

Robin Wellmann

Examples

```
data(PedigWithErrors)
data(Phen)
keep  <- Phen$Indiv
Pedig <- prePed(PedigWithErrors, keep=keep, thisBreed="Hinterwaelder", lastNative=1970)
pKinatN <- pedIBDatN(Pedig, thisBreed="Hinterwaelder", keep.only=keep, nGen=6)

#Number of Migrant Founders: 237
#Number of Native  Founders: 150
#Individuals in Pedigree   : 1658
#Native effective size     : 49.5

## Mean kinship at native segments:
pKinatN$mean
#[1] 0.0776925

## Note that this can not be computed as mean(pKinatN$of).

## Results for individuals:
pKinatN$of <- pKinatN$Q1/pKinatN$Q2
pKinatN$of["276000812497583","276000812496823"]
#[1] 0.05941229
```

pedIBDorM *Calculates Kinships taking Allele Origin into Account*

Description

Calculates the **ped**igree based probability of alleles to be **IBD** (identical by descent) **or M**igrant alleles: For each pair of individuals the probability is computed that two alleles taken at random are IBD or are migrant alleles.

Usage

```
pedIBDorM(Pedig, thisBreed=NA, keep.only=NULL, keep=keep.only)
```

Arguments

`Pedig`	Data frame containing the Pedigree. The data frame has columns (1) Individual, (2) Sire, (3) Dam, (4) Sex, and (5) Breed. Missing parents are coded as NA. Both parents must either be missing or present. If this is not the case use prePed.
`thisBreed`	Name of the breed in column (5) of the pedigree for which the kinships are to be computed.
`keep`	If `keep` is provided then kinships are computed only for these animals and their ancestors.
`keep.only`	If `keep.only` is provided then kinships are computed only for these animals.

Details

Computation of modified pedigree based kinship matrices taking allele origin into account.

A native founder is an individual with unkown parents belonging to `thisBreed`. A migrant is an individual with unkown parents not belonging to `thisBreed`.

Value

A list with the following components:

`pedIBDorM`	Matrix containing for individuals i and j the probability that two alleles chosen from the individuals are IBD or at least one of them is a migrant allele (only computed if 1 is in `method`)
`pedIBDorMM`	Matrix containing for individuals i and j the probability that two alleles chosen from the individuals are IBD or both are migrant alleles (only computed if 2 is in `method`)

Author(s)

Robin Wellmann

Examples

```
data(PedigWithErrors)
data(Phen)
keep  <- Phen$Indiv
Pedig <- prePed(PedigWithErrors, keep=keep, thisBreed="Hinterwaelder", lastNative=1970)
Kin   <- pedIBDorM(Pedig, thisBreed="Hinterwaelder", keep.only=keep)

mean(Kin$pedIBDorM)
#[1] 0.8201792
mean(Kin$pedIBDorMM)
#[1] 0.335358
```

PedigWithErrors *Pedigree of Hinterwald cattle*

Description

This data set gives the pedigree of Hinterwald cattle with some artificially introduced errors and simulated breeding values.

Usage

```
data(PedigWithErrors)
```

Format

A data frame with columns `Indiv` (individual ID), `Sire`, `Dam`, `Sex`, `Breed`, `Born` with year of birth, and column `BV` with simulated breeding values.

---

pedInbreeding *Calculates Pedigree Based Inbreeding*

Description

Calculates Pedigree Based Inbreeding

Usage

```
pedInbreeding(Pedig)
```

Arguments

`Pedig`	Data frame containing the Pedigree with the first 3 columns being `Indiv` (individual ID), `Sire`, and `Dam`, which is usually obtained with function prePed. Missing parents are coded as NA.

Details

Computation of pedigree based inbreeding. This function is a wrapper function for pedigree from package pedigree.

Value

A data frame with column Indiv containing the individual IDs and column Inbr containing the inbreeding coefficients.

Author(s)

Robin Wellmann

Examples

```
data(PedigWithErrors)
data(Phen)
keep  <- Phen$Indiv
Pedig <- prePed(PedigWithErrors, keep=keep)
Res   <- pedInbreeding(Pedig)
mean(Res$Inbr[Res$Indiv %in% keep])
#[1] 0.01943394
```

---

pedplot *Plots a Pedigree*

Description

Plots a pedigree

Usage

```
pedplot(Pedig, affected=NULL, status=NULL, label="Indiv", ...)
```

Arguments

Pedig	Data frame containing the pedigree with columns Indiv (indivdual ID), Sire, Dam, and Sex. Use subPed to ensure that the pedigree is in the correct format.
affected	Logical vector indicating for each individual if its symbol should be plotted in colour. The default NULL means that the individuals in column keep of data frame Pedig are plotted in colour (if present).
status	Logical vector indicating for each individual if its symbol in the plot should be crossed out. The default NULL means that animals from other breeds than those plotted in colour are crossed out.
label	Character vector containing the columns of data frame Pedig to be used as labels.
...	Options passed to the underlying function plot.pedigree from package kinship2.

Details

This function plots a pedigree. If data frame `Pedig` has logical column `keep` then the default values mean that the symbols of these animals are plotted in color and for animals from other breeds the symbol is crossed out.

Value

An invisible list returned by the underlying function plot.pedigree from package `kinship2`.

Author(s)

Robin Wellmann

Examples

```
data(PedigWithErrors)

sPed  <- subPed(PedigWithErrors, keep="276000810087543", prevGen=3, succGen=2)
pedplot(sPed, mar=c(2,4,2,4), label=c("Indiv", "Born", "Breed"), cex=0.4)
```

Phen *Simulated Phenotypes of Hinterwald Cattle*

Description

A data frame simulated breeding values of some Hinterwald cattle with offspring born in 2006 or 2007.

Usage

```
data(Phen)
```

Format

A data frame with individual IDs (`Indiv`), sexes (`Sex`), breeding values (`BV`), and native contribution (`NC`).

plot.HaploFreq *Plots Frequencies of Haplotype Segments in Specified Breeds*

Description

For a particular haplotype from thisBreed and each marker m the frequency of the segment containing marker m in a specified reference breed is plotted.

Usage

```
## S3 method for class 'HaploFreq'
plot(x, ID=1, hap=1, refBreed=NULL, Chr=NULL, show.maxFreq=FALSE, ...)
```

Arguments

x	This is either an R-Object obtained with function haplofreq or a list obtained with function freqlist.
ID	Either the ID of the animal from this breed to be plotted, or the position of the animal in R-Object x.
hap	Number of the haplotype to be plotted (1 or 2)
refBreed	Breed name. The frequencies the haplotype segments have in this reference breed will be plotted. Parameter refBreeds="others" means that the maximum frequency will be plotted the segments have in other breeds.
Chr	Vector with chromosomes to be plotted. The default means that all chromosomes will be plotted.
show.maxFreq	If show.maxFreq=TRUE then a peak of the grey curve means that a haplotype segment exist in the breed which has high frequency in one of the reference breeds. This frequency is shown. The default is FALSE.
...	Arguments to be passed to methods, such as graphical parameters.

Details

For a particular haplotype from thisBreed and each marker m from chromosomes Chr the frequency of the segment containing marker m in reference breed refBreed is plotted (red line), as well as the maximum frequency the segment has in one of the evaluated breeds (black line), and the maximum frequency a segment from thisBreed has in one of the evaluated breeds (grey area, if show.maxFreq=TRUE).

Value

No return value, called for plotting.

Author(s)

Robin Wellmann

Examples

```
data(map)
data(Cattle)
dir   <- system.file("extdata", package="optiSel")
files <- paste(dir, "/Chr", 1:2, ".phased", sep="")

Freq <- freqlist(
  haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="Rotbunt",   minSNP=20),
  haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="Holstein",  minSNP=20),
  haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="Fleckvieh", minSNP=20)
  )

names(Freq)

plot(Freq, ID=1, hap=2, refBreed="Rotbunt")

Freq <- haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="others",  minSNP=20)

plot(Freq, ID=1, hap=2)
plot(Freq, ID=1, hap=2, show.maxFreq=TRUE)

Freq <- haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="Angler",  minSNP=20)
plot(Freq, ID=1, hap=2)
```

---

prePed *Prepares a Pedigree*

Description

Prepares a pedigree by sorting and adding founders and pruning the pedigree.

Usage

```
prePed(Pedig, keep=NULL, thisBreed=NA, lastNative=NA, addNum=FALSE)
```

Arguments

`Pedig`	Data frame containing the pedigree where the first 3 columns correspond to: Individual ID, Sire, and Dam. More columns can be passed in the `Pedig` argument including columns named `Sex`, `Breed` (with breed names), and `Born` (with years of birth or generation numbers). Missing parents are coded as NA, 0, or "0".
`keep`	Vector with IDs of individuals, or `NULL`, or a logical vector indicating the individuals to be kept. If this parameter is not `NULL`, then only these individuals and their ancestors will be kept in the pedigree.
`thisBreed`	Name of the breed.
`lastNative`	Last year of birth for which individuals with unknown pedigree are considered native.

addNum — If TRUE, then columns with IDs of individuals, sires, and dams in integer form will be added.

Details

This function takes a pedigree, adds missing founders, and sorts the pedigree. If parameter keep contains IDs of individuals then only these individuals and their ancestors will be kept in the pedigree.

If the pedigree contains loops, then the loops will be broken by setting the parents of one animal in each loop to NA.

If the pedigree contains columnn Sex then the sexes will be recoded as 'male' and 'female'. Missing sexes will be determined from pedigree structure if possible.

If the pedigree contains column Breed then for ancestors with missing breed the breed name is estimated. If parameter lastNative is not NA then for each animal with one missing parent an imaginary founder is added to the pedigree in order to enable classifying the breed names of all founders as follows: In general animals with missing breed are assumed to have the same breed as most of their offspring. But there is one exception: For founders belonging to thisBreed who are born after lastNative the breed name will be set to "unknown". Moreover for founders from thisBreed with unknown year of birth the breed name will be set to "unknown" if all their descendants are born after lastNative+I.

Value

Data frame containing the pedigree with columns:

Indiv	Character column with IDs of the individuals
Sire	Character column with IDs of the sires
Dam	Character column with IDs of the dams
Sex	Character column with sexes of the individuals denoted as "male" and "female".
Breed	Character column with adjusted breed names of the individuals (only if Pedig has column Breed.).
Born	Numeric column with Year-of-Birth of the individuals (only if Pedig has column Born.).
I	Numeric column with the average age of the parents when the respective individual was born (only if Pedig has column Born.).
numIndiv	Numeric IDs of the individuals, which are equal to the row numbers (only if addNum=TRUE).
numSire	Numeric IDs of the sires (only if addNum=TRUE).
numDam	Numeric IDs of the dams (only if addNum=TRUE).
Offspring	Logical column indicating the individuals with offspring.

Author(s)

Robin Wellmann

Examples

```
data(PedigWithErrors)
Pedig <- prePed(PedigWithErrors)

tail(Pedig)
hist(Pedig$I, freq=FALSE, ylim=c(0,0.2))
```

`read.indiv` *Reads Individual IDs from a Genotype File*

Description

Reads individual IDs from a genotype file.

Usage

```
read.indiv(file, skip=NA, cskip=NA)
```

Arguments

`file`	Name of the genotype file.
`skip`	Take line `skip+1` of the genotype files as the row with column names. By default, the number is determined automatically.
`cskip`	Take column `cskip+1` of the genotype files as the first column with genotypes. By default, the number is determined automatically.

Details

Reading individual IDs from phased marker files.

Marker file format: Each marker file containing phased genotypes has a header and no row names. Cells are separated by blank spaces. The number of rows is equal to the number of markers from the respective chromosome and the markers are in the same order as in the `map`. The first `cskip` columns are ignored. The remaining columns contain genotypes of individuals written as two alleles separated by a character, e.g. A/B, 0/1, A|B, A B, or 0 1. The same two symbols must be used for all markers. Column names are the IDs of the individuals. If the blank space is used as separator then the ID of each individual should repeated in the header to get a regular delimited file. The columns to be skipped and the individual IDs must have no white spaces. The name of each file must contain the chromosome name as specified in the `map` in the form `"ChrNAME."`, e.g. `"Breed2.Chr1.phased"`.

Value

Vector with the IDs of the individuals.

Author(s)

Robin Wellmann

Examples

```
data(Cattle)

dir   <- system.file("extdata", package = "optiSel")
file  <- file.path(dir, "Chr1.phased")
ID    <- read.indiv(file)

identical(Cattle$Indiv, ID)
#[1] TRUE
```

`sampleIndiv` *Sample Individuals from Pedigree*

Description

Sampling Individuals from a Pedigree.

Usage

```
sampleIndiv(Pedig, from="Born", each=100)
```

Arguments

`Pedig`	Pedigree with column `Indiv` and the column specified in parameter `from`.
`from`	Column name. From each cohort specified in this column (e.g. year of birth), the number of individuals specified in parameter `each` is sampled. If a cohort contains less individuals, then all individuals are chosen.
`each`	Number of individuals to be sampled from each cohort.

Details

From each cohort, a specified number of individuals will be sampled. If a cohort contains less individuals, then all individuals are sampled. This may be needed for estimating population specific parameters from a subset of a large pedigree to reduce computation time.

Value

Character vector containing the IDs of the individuals.

Author(s)

Robin Wellmann

Examples

```
data("PedigWithErrors")
set.seed(1)
Pedig <- prePed(PedigWithErrors)
use   <- Pedig$Breed=="Hinterwaelder"
keep  <- sampleIndiv(Pedig[use, ], from="Born", each=5)
keep
```

---

segBreedComp *Calculates the Segment-Based Breed Composition of Individuals*

Description

Calculates the **seg**ment based **Breed Comp**osition: For every individual the breed composition is estimated, including the genetic contribution from native ancestors.

Usage

```
segBreedComp(Native, map, unitP="Mb")
```

Arguments

Native	This parameter is either (1) Mx(2N) logical matrix, with TRUE, if the segment containing the SNP is considered native, and FALSE otherwise. The row names are the marker names, and the non-unique column names are the IDs of the individuals. The matrix is typically computed from component freq of the output from function haplofreq. or (2) Mx(2N) character matrix, with components being the first characters of the names of the breeds in which the respective segment has maximum frequency. Segments considered native are coded as '1'. The row names are the marker names, and the non-unique column names are the IDs of the individuals. The matrix is typically component match from the output of function haplofreq. or (3) Vector with file names. The files contain for every SNP and for each haplotype 1 if the segment containing the SNP is considered native. Otherwise it is the first letter of the name of the breed in which the segment has maximum frequency. These files are typically created by function haplofreq. There is one file per chromosome and file names must contain the chromosome name as specified in the map in the form "ChrNAME.", e.g. "Breed2.Chr1.nat".
map	Data frame providing the marker map with columns including marker name 'Name', chromosome number 'Chr', and possibly the position on the chromosome in Mega base pairs 'Mb', and the position in centimorgan 'cM'. The markers must be in the same order as in in Native.

`unitP` The unit for measuring the proportion of the genome included in native segments. Possible units are the number of marker SNPs included in shared segments (`'SNP'`), the number of Mega base pairs (`'Mb'`), and the total length of the shared segments in centimorgan (`'cM'`). In the last two cases the map must include columns with the respective names.

Details

For every individual the breed composition is computed, including the genetic contribution from native ancestors (native contribution). The native contribution is the proportion of the genome belonging to segments whose frequency is smaller than a predefined value in all other breeds.

Additionally, for each introgressed breed, the proportion of the genome of each individual is computed that is non-native and has maximum frequency in the respective breed (not if option (1) is used).

Value

Data frame with the number of rows being the number of individuals. The columns are

`Indiv`	IDs of the individuals,
`native`	Genetic contributions from native ancestors,
`...`	Contributions from other breeds.

Author(s)

Robin Wellmann

Examples

```
data(map)
data(Cattle)
dir     <- system.file("extdata", package = "optiSel")
GTfiles <- file.path(dir, paste("Chr", unique(map$Chr), ".phased", sep=""))
Haplo   <- haplofreq(GTfiles, Cattle, map, thisBreed="Angler", minSNP=20, minL=1.0)
Comp    <- segBreedComp(Haplo$freq<0.01, map)
mean(Comp$native)
#[1] 0.3853432

Comp <- segBreedComp(Haplo$match, map)
apply(Comp[, -1], 2, mean)

## Reading native segments from files:

wdir <- file.path(tempdir(), "HaplotypeEval")
file <- haplofreq(GTfiles, Cattle, map, thisBreed="Angler", minSNP=20,
          minL=1.0, ubFreq=0.01, what="match", w.dir=wdir)
Comp <- segBreedComp(file$match, map)
head(Comp)

apply(Comp[, -1], 2, mean)
#    native          F          H          R
```

```
#0.38534317 0.05503451 0.25986508 0.29975724

#unlink(wdir, recursive = TRUE)
```

---

segIBD *Calculates the Segment Based Kinship Matrix*

Description

Segment based probability of alleles to be **IBD** (identical by descent): For each pair of individuals the probability is computed that two alleles taken at random position from randomly chosen haplotypes belong to a shared segment.

Usage

```
segIBD(files, map, minSNP=20, minL=1.0, unitP="Mb", unitL="Mb",
   a=0.0, keep=NULL, skip=NA, cskip=NA, cores=1, quiet=FALSE)
```

Arguments

files	This parameter is either (1) A vector with names of phased marker files, one file for each chromosome, or (2) A list with two components. Each component is a vector with names of phased marker files, one file for each chromosome. Each components corresponds to a different set of individuals. This enables to compute the kinship between individuals stored in two different files. File names must contain the chromosome name as specified in the map in the form "ChrNAME.", e.g. "Breed2.Chr1.phased". The required format of the marker files is described under Details.
map	Data frame providing the marker map with columns including marker name 'Name', chromosome number 'Chr', and possibly the position on the chromosome in mega base pairs 'Mb', and the position in centimorgan 'cM'. (The position in base pairs could result in an integer overflow.) The order of the markers must be the same as in the files.
minSNP	Minimum number of marker SNPs included in a segment.
minL	Minimum length of a segment in unitL (e.g. in cM or Mb).
unitP	The unit for measuring the proportion of the genome included in shared segments. Possible units are the number of marker SNPs included in shared segments ('SNP'), the number of mega base pairs ('Mb'), and the total length of the shared segments in centimorgan ('cM'). In the last two cases the map must include columns with the respective names.

unitL	The unit for measuring the length of a segment. Possible units are the number of marker SNPs included in the segment ('SNP'), the number of mega base pairs ('Mb'), and the genetic distances between the first and the last marker in centimorgan ('cM'). In the last two cases the map must include columns with the respective names.
a	The Function providing the weighting factor for each segment is w(x)=x*x/(a+x*x). The parameter of the function is the length of the segment in unitL. The default value a=0.0 implies no weighting, whereas a>0.0 implies that old inbreeding has less influence on the result than new inbreeding.
keep	If keep is a vector containing IDs of individuals then kinships will be computed only for these individuals. The default keep=NULL means that kinship will be computed for all individuals included in the files.
skip	Take line skip+1 of the files as the row with column names. By default, the number is determined automatically.
cskip	Take column cskip+1 of the files as the first column with genotypes. By default, the number is determined automatically.
cores	Number of cores to be used for parallel processing of chromosomes. By default one core is used. For cores=NA the number of cores will be chosen automatically. Using more than one core increases execution time if the function is already fast.
quiet	Should console output be suppressed?

Details

For each pair of individuals the probability is computed that two SNPs taken at random position from randomly chosen haplotypes belong to a shared segment.

Genotype file format: Each file containing phased genotypes has a header and no row names. Cells are separated by blank spaces. The number of rows is equal to the number of markers from the respective chromosome and the markers are in the same order as in the map. The first cskip columns are ignored. The remaining columns contain genotypes of individuals written as two alleles separated by a character, e.g. A/B, 0/1, A|B, A B, or 0 1. The same two symbols must be used for all markers. Column names are the IDs of the individuals. If the blank space is used as separator then the ID of each individual should repeated in the header to get a regular delimited file. The columns to be skipped and the individual IDs must have no white spaces.

Value

NxN segment-based kinship matrix with N being the number of individuals.

Author(s)

Robin Wellmann

References

de Cara MAR, Villanueva B, Toro MA, Fernandez J (2013). Using genomic tools to maintain diversity and fitness in conservation programmes. Molecular Ecology. 22: 6091-6099

Examples

```
data(map)
dir   <- system.file("extdata", package = "optiSel")
files <- file.path(dir, paste("Chr", unique(map$Chr), ".phased", sep=""))
f     <- segIBD(files, map, minSNP=15, minL=1.0)
mean(f)
#[1] 0.05677993


f     <- segIBD(files, map, minSNP=15, minL=1.0, cores=NA)
mean(f)
#[1] 0.05677993



## Multidimensional scaling of animals:
## (note that only few markers are used)

data(Cattle)
library("smacof")
D     <- sim2dis(f, 4)
color <- c(Angler="red", Rotbunt="green", Fleckvieh="blue", Holstein="black")
col   <- color[as.character(Cattle$Breed)]
Res   <- smacofSym(D, itmax = 5000, eps = 1e-08)
plot(Res$conf, pch=18, col=col, main="Multidimensional Scaling", cex=0.5)
mtext(paste("segIBD Stress1 = ", round(Res$stress,3)))
```

---

segIBDandN *Calculates Probabilities that Alleles belong to a Shared Native Segment*

Description

Calculates the **seg**ment based probability of alleles to be **IBD** (identical by descent) **and N**ative: For each pair of individuals the probability is computed that two SNPs taken at random position from randomly chosen haplotypes belong to a shared segment and are native.

Usage

```
segIBDandN(files, Native, map, minSNP=20, unitP="Mb", minL=1.0,
   unitL="Mb", a=0.0, keep=NULL, skip=NA, cskip=NA, cores=1, quiet=FALSE)
```

Arguments

files	Vector with names of the phased marker files, one file for each chromosome. The required format is described under Details. File names must contain the chromosome name as specified in the map in the form "ChrNAME.", e.g. "Breed2.Chr1.phased".

`Native` This parameter is either

(1) Mx(2N) indicator matrix, with 1, if the segment containing the SNP is considered native, and 0 otherwise. The row names are the marker names, and the non-unique column names are the IDs of the individuals. The matrix is typically computed from the output of function haplofreq.

or

(2) Vector with file names. The files contain for every SNP and for each haplotype from this breed 1 if the segment containing the SNP is considered native. These files are typically created by function haplofreq. There is one file per chromosome and file names must contain the chromosome name as specified in the `map` in the form `"ChrNAME."`, e.g. `"Breed2.Chr1.nat"`.

`map` Data frame providing the marker map with columns including marker name `'Name'`, chromosome number `'Chr'`, and possibly the position on the chromosome in mega base pairs `'Mb'`, and the position in centimorgan `'cM'`. The markers must be in the same order as in `files` and in `Native`.

`minSNP` Minimum number of marker SNPs included in a segment.

`unitP` The unit for measuring the proportion of the genome included in shared segments. Possible units are the number of marker SNPs included in shared segments (`'SNP'`), the number of Mega base pairs (`'Mb'`), and the total length of the shared segments in centiMorgan (`'cM'`). In the last two cases the map must include columns with the respective names.

`minL` Minimum length of a segment in `unitL` (e.g. in cM or Mb).

`unitL` The unit for measuring the length of a segment. Possible units are the number of marker SNPs included in the segment (`'SNP'`), the number of Mega base pairs (`'Mb'`), and the genetic distances between the first and the last marker in centiMorgan (`'cM'`). In the last two cases the map must include columns with the respective names.

`a` The Function providing the weighting factor for each segment is w(x)=x*x/(a+x*x). The parameter of the function is the length of the segment in `unitL`. The default value `a=0.0` implies no weighting, whereas `a>0.0` implies that old inbreeding has less influence on the result than new inbreeding.

`keep` Vector with IDs of individuals (from this breed) for which the probabilities are to be computed. By default, they will be computed for all individuals included in `Native`.

`skip` Take line `skip+1` of the genotype files as the line with column names. By default, the number is determined automatically.

`cskip` Take column `cskip+1` of the genotype files as the first column with genotypes. By default, the number is determined automatically.

`cores` Number of cores to be used for parallel processing of chromosomes. By default one core is used. For `cores=NA` the number of cores will be chosen automatically. Using more than one core increases execution time if the function is already fast.

`quiet` Should console output be suppressed?

Details

For each pair of individuals the probability is computed that two SNPs taken at random position from randomly chosen haplotypes belong to a shared segment and are native. That is, they are not introgressed from other breeds.

Genotype file format: Each file containing phased genotypes has a header and no row names. Cells are separated by blank spaces. The number of rows is equal to the number of markers from the respective chromosome and the markers are in the same order as in the `map`. The first `cskip` columns are ignored. The remaining columns contain genotypes of individuals written as two alleles separated by a character, e.g. A/B, 0/1, A|B, A B, or 0 1. The same two symbols must be used for all markers. Column names are the IDs of the individuals. If the blank space is used as separator then the ID of each individual should repeated in the header to get a regular delimited file. The columns to be skipped and the individual IDs must have no white spaces. The name of each file must contain the chromosome name as specified in the `map` in the form `"ChrNAME."`, e.g. `"Breed2.Chr1.phased"`.

Value

`NxN` matrix with `N` being the number of individuals from this breed included in all files (and in parameter `keep`).

Author(s)

Robin Wellmann

Examples

```
data(map)
data(Cattle)
dir    <- system.file("extdata", package = "optiSel")
GTfile <- file.path(dir, paste("Chr", unique(map$Chr), ".phased", sep=""))
Freq  <- haplofreq(GTfile, Cattle, map, thisBreed="Angler", refBreeds="others", minSNP=20)$freq

fIBDN  <- segIBDandN(GTfile, Freq<0.01, map=map, minSNP=20)
mean(fIBDN)
#[1] 0.01032261


fIBDN  <- segIBDandN(GTfile, Freq<0.01, map=map, minSNP=20, cores=NA)
mean(fIBDN)
#[1] 0.01032261


## using files:

wdir   <- file.path(tempdir(),"HaplotypeEval")
chr    <- unique(map$Chr)
GTfile <- file.path( dir, paste("Chr", chr, ".phased",     sep=""))
file  <- haplofreq(GTfile, Cattle, map, thisBreed="Angler", minSNP=20, ubFreq=0.01, w.dir=wdir)

fIBDN  <- segIBDandN(GTfile, file$match, map=map, minSNP=20)
mean(fIBDN)
```

```
#[1] 0.01032261

fIBDN  <- segIBDandN(GTfile, file$match, map=map, minSNP=20, cores=NA)
mean(fIBDN)
#[1] 0.01032261


#unlink(wdir, recursive = TRUE)
```

---

segIBDatN *Segment-Based Kinship at Native Alleles.*

Description

Calculates the kinship at native alleles, which is the segment based probability of native alleles to be IBD.

Usage

```
segIBDatN(files, phen, map, thisBreed, refBreeds="others", ubFreq=0.01, minSNP=20,
  unitP="Mb", minL=1.0, unitL="Mb", a=0.0, keep=NULL, lowMem=TRUE,
  skip=NA, cskip=NA, cores=1, quiet=FALSE)
```

Arguments

files	This can be a character vector with names of the phased marker files, one file for each chromosome. Alternatively files can be a list with the following components: a) hap.thisBreed: A character vector with names of the phased marker files for the individuals from thisBreed, one file for each chromosome. b) hap.refBreeds: A character vector with names of the phased marker files for the individuals from the reference breeds (refBreeds), one file for each chromosome. If this component is missing, then it is assumed that the haplotypes of these animals are also included in hap.thisBreed. c) match: If present, a character vector with file names containing the origin of the marker alleles. The files are typically created with function haplofreq. If this vector is missing, then the default method is used to estimate the origins. File names must contain the chromosome name as specified in the map in the form "ChrNAME.", e.g. "Breed2.Chr1.phased". The required format of the marker files is described under Details.
phen	Data frame containing the ID (column "Indiv"), breed name (column "Breed"), and sex (column Sex) of each individual.

map	Data frame providing the marker map with columns including marker name 'Name', chromosome number 'Chr', and possibly the position on the chromosome in Mega base pairs 'Mb', and the position in centimorgan 'cM'. (The position in base pairs could result in an integer overflow). The order of the markers must bethe same as in the files.
thisBreed	Breed name: Results will be computed for individuals from thisBreed.
refBreeds	Vector containing names of genotyped breeds. A segment is considered native if its frequency is smaller than ubFreq in all refBreeds. The default "others" means that all genotyped breeds except thisBreed are considered.
ubFreq	A segment is considered native if its frequency is smaller than ubFreq in all reference breeds.
minSNP	Minimum number of marker SNPs included in a segment.
unitP	The unit for measuring the proportion of the genome included in native segments. Possible units are the number of marker SNPs included in shared segments ('SNP'), the number of Mega base pairs ('Mb'), and the total length of the shared segments in centimorgan ('cM'). In the last two cases the map must include columns with the respective names.
minL	Minimum length of a segment in unitL (e.g. in cM).
unitL	The unit for measuring the length of a segment. Possible units are the number of marker SNPs included in the segment ('SNP'), the number of Mega base pairs ('Mb'), and the genetic distances between the first and the last marker in centimorgan ('cM'). In the last two cases the map must include columns with the respective names.
a	The function providing the weighting factor for each segment is w(x)=x*x/(a+x*x). The parameter of the function is the length of the segment in unitL. The default value a=0.0 implies no weighting, whereas a>0.0 implies that old inbreeding has less influence on the result than new inbreeding.
keep	Subset of the IDs of the individuals from data frame phen (including individuals from other breeds) or a logical vector indicating the animals in data frame phen that should be used. By default all individuals included in phen will be used.
lowMem	If lowMem=TRUE then temporary files will be created and deleted.
skip	Take line skip+1 of the genotype files as the row with column names. By default, the number is determined automatically.
cskip	Take column cskip+1 of the genotype files as the first column with genotypes. By default, the number is determined automatically.
cores	Number of cores to be used for parallel processing of chromosomes. By default one core is used. For cores=NA the number of cores will be chosen automatically. Using more than one core increases execution time if the function is already fast.
quiet	Should console output be suppressed?

Details

Calculates a list containing matrices needed to compute segment based kinships at native alleles, defined as the conditional probability that two randomly chosen alleles are IBD, given that both

originate from native ancestors. An allele is considered to originate from a native ancesor if the segment containing the allele has low frequency in all reference breeds.

The kinship at native alleles between individuals i and j is `Q1[i,j]/Q2[i,j]`.

The mean kinship at native alleles in the offspring is `(x'Q1x+d1)/(x'Q2x+d2)`, where `x` is the vector with genetic contributions of the selection candidates.

Genotype file format: Each file containing phased genotypes has a header and no row names. Cells are separated by blank spaces. The number of rows is equal to the number of markers from the respective chromosome and the markers are in the same order as in the `map`. The first `cskip` columns are ignored. The remaining columns contain genotypes of individuals written as two alleles separated by a character, e.g. A/B, 0/1, A|B, A B, or 0 1. The same two symbols must be used for all markers. Column names are the IDs of the individuals. If the blank space is used as separator then the ID of each individual should repeated in the header to get a regular delimited file. The columns to be skipped and the individual IDs must have no white spaces. The name of each file must contain the chromosome name as specified in the `map` in the form `"ChrNAME."`, e.g. `"Breed2.Chr1.phased"`.

Value

A list of class `ratioFun` including components:

`Q1`	matrix with `Q1[i,j]` = Probability that two alleles chosen from individuals i and j are IBD and are native.
`Q2`	matrix with `Q2[i,j]` = Probability that two alleles chosen from individuals i and j are both native.
`d1`	The value by which the probability that two alleles chosen from the offspring are IBD and native increases due to genetic drift.
`d2`	The value by which the probability that two alleles chosen from the offspring are native increases due to genetic drift.
`id`	IDs of the individuals for which the probabilites have been computed.
`mean`	Mean kinship at native alleles of the specified individuals. Note that `1-mean` is the genetic diversity at native segments of the specified individuals from `thisBreed`

Author(s)

Robin Wellmann

Examples

```
data(map)
data(Cattle)
dir     <- system.file("extdata", package = "optiSel")
files   <- paste(dir, "/Chr", 1:2, ".phased", sep="")
sKinatN <- segIBDatN(files, Cattle, map, thisBreed="Angler",
                     ubFreq=0.01, minL=1.0, lowMem=FALSE)

## Mean kinship at native segments:
sKinatN$mean
#[1] 0.06695171
```

```
## Note that this can not be computed as mean(sKinatN$of).


## Results for individuals:
sKinatN$of <- sKinatN$Q1/sKinatN$Q2
sKinatN$of["Angler1","Angler5"]
#[1] 0.4394066

## Use temporary files to reduce working memory:

sKinatN <- segIBDatN(files, Cattle, map, thisBreed="Angler", ubFreq=0.01,  minL=1.0)

## Mean kinship at native segments:
sKinatN$mean
#[1] 0.06695171
```

`segInbreeding` *Calculates Segment Based Inbreeding*

Description

Segment based **Inbreeding**: For each individual the probability is computed that the paternal allele and the maternal allele, sampled from random position, belong to a shared segment (i.e. a run of homozygosity, ROH). The arguments are the same as for function segIBD.

Usage

```
segInbreeding(files, map, minSNP=20, minL=1.0, unitP="Mb", unitL="Mb",
   a=0.0, keep=NULL, skip=NA, cskip=NA, quiet=FALSE)
```

Arguments

`files`	This parameter is either (1) A vector with names of phased marker files, one file for each chromosome, or (2) A list with two components. Each component is a vector with names of phased marker files, one file for each chromosome. Each components corresponds to a different set of individuals. File names must contain the chromosome name as specified in the `map` in the form `"ChrNAME."`, e.g. `"Breed2.Chr1.phased"`. The required format of the marker files is described under `Details`.
`map`	Data frame providing the marker map with columns including marker name `'Name'`, chromosome number `'Chr'`, and possibly the position on the chromosome in Mega base pairs `'Mb'`, and the position in centimorgan `'cM'`. (The position in base pairs could result in an integer overflow.) The order of the markers must be the same as in the files.

minSNP	Minimum number of marker SNPs included in a segment.
minL	Minimum length of a segment in unitL (e.g. in cM or Mb).
unitP	The unit for measuring the proportion of the genome included in shared segments. Possible units are the number of marker SNPs included in shared segments ('SNP'), the number of Mega base pairs ('Mb'), and the total length of the shared segments in centimorgan ('cM'). In the last two cases the map must include columns with the respective names.
unitL	The unit for measuring the length of a segment. Possible units are the number of marker SNPs included in the segment ('SNP'), the number of Mega base pairs ('Mb'), and the genetic distances between the first and the last marker in centimorgan ('cM'). In the last two cases the map must include columns with the respective names.
a	The Function providing the weighting factor for each segment is w(x)=x*x/(a+x*x). The parameter of the function is the length of the segment in unitL. The default value a=0.0 implies no weighting, whereas a>0.0 implies that old inbreeding has less influence on the result than new inbreeding.
keep	If keep is a vector containing IDs of individuals then inbreeding will be computed only for these individuals. The default keep=NULL means that inbreeding will be computed for all individuals included in the files.
skip	Take line skip+1 of the files as the row with column names. By default, the number is determined automatically.
cskip	Take column cskip+1 of the files as the first column with genotypes. By default, the number is determined automatically.
quiet	Should console output be suppressed?

Details

For each pair of individuals the probability is computed that two SNPs taken at random position from randomly chosen haplotypes belong to a shared segment.

Genotype file format: Each file containing phased genotypes has a header and no row names. Cells are separated by blank spaces. The number of rows is equal to the number of markers from the respective chromosome and the markers are in the same order as in the map. The first cskip columns are ignored. The remaining columns contain genotypes of individuals written as two alleles separated by a character, e.g. A/B, 0/1, A|B, A B, or 0 1. The same two symbols must be used for all markers. Column names are the IDs of the individuals. If the blank space is used as separator then the ID of each individual should repeated in the header to get a regular delimited file. The columns to be skipped and the individual IDs must have no white spaces.

Value

A data frame with column Indiv containing the individual IDs and column Inbr containing the inbreeding coefficients.

Author(s)

Robin Wellmann

References

de Cara MAR, Villanueva B, Toro MA, Fernandez J (2013). Using genomic tools to maintain diversity and fitness in conservation programmes. Molecular Ecology. 22: 6091-6099

Examples

```
data(map)
data(Cattle)
dir   <- system.file("extdata", package = "optiSel")
files <- file.path(dir, paste("Chr", 1:2, ".phased", sep=""))
f     <- segInbreeding(files, map, minSNP=20, minL=2.0)

Cattle2 <- merge(Cattle, f, by="Indiv")
tapply(Cattle2$Inbr, Cattle2$Breed, mean)
#    Angler  Fleckvieh   Holstein    Rotbunt
#0.03842552 0.05169508 0.12431393 0.08386849

boxplot(Inbr~Breed, data=Cattle2, ylim=c(0,1), main="Segment Based Inbreeding")

fIBD  <- segIBD(files, map, minSNP=20, minL=2.0)
identical(rownames(fIBD), f$Indiv)
#[1] TRUE

range(2*diag(fIBD)-1-f$Inbr)
#[1] -2.220446e-16  2.220446e-16
```

---

segN *Calculates Probabilities of Alleles to belong to Native Segments*

Description

Segment based probability of alleles to be **N**ative: For each pair of individuals the probability is computed that two SNPs taken at random position from randomly chosen haplotypes both belong to native segments.

Usage

```
segN(Native, map, unitP="Mb", keep=NULL, cores=1, quiet=FALSE)
```

Arguments

Native	This parameter is either (1) Mx(2N) indicator matrix, with 1, if the segment containing the SNP is considered native, and 0 otherwise. The row names are the marker names, and the non-unique column names are the IDs of the individuals. The matrix is typically computed from the output of function haplofreq. or

(2) Vector with file names. The files contain for every SNP and for each haplotype from this breed 1 if the segment containing the SNP is considered native. These files are typically created by function haplofreq. There is one file per chromosome and file names must contain the chromosome name as specified in the map in the form "ChrNAME.", e.g. "Breed2.Chr1.nat".

`map` Data frame providing the marker map with columns including marker name 'Name', chromosome number 'Chr', and possibly the position on the chromosome in Mega base pairs 'Mb', and the position in centiMorgan 'cM'. The markers must be in the same order as in Native.

`unitP` The unit for measuring the proportion of the genome included in native segments. Possible units are the number of marker SNPs included in shared segments ('SNP'), the number of Mega base pairs ('Mb'), and the total length of the shared segments in centimorgan ('cM'). In the last two cases the map must include columns with the respective names.

`keep` Vector with IDs of individuals (from this breed) for which the probabilities are to be computed. By default, they will be computed for all individuals included in Native.

`cores` Number of cores to be used for parallel processing of chromosomes. By default one core is used. For cores=NA the number of cores will be chosen automatically. Using more than one core increases execution time if the function is already fast.

`quiet` Should console output be suppressed?

Details

For each pair of individuals the probability is computed that two SNPs taken at random position from randomly chosen haplotypes both belong to native segments. That is, they are not introgressed from other breeds.

Value

NxN matrix with N being the number of genotyped individuals from this breed (which are also included in vector keep).

Author(s)

Robin Wellmann

Examples

```
data(map)
data(Cattle)
dir   <- system.file("extdata", package = "optiSel")
files <- file.path(dir, paste("Chr", unique(map$Chr), ".phased", sep=""))
Freq <- haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="others", minSNP=20)$freq
fN   <- segN(Freq<0.01, map)
mean(fN)
#[1] 0.15418
```

```
fN   <- segN(Freq<0.01, map, cores=NA)
mean(fN)
#[1] 0.15418



## using files:

wdir   <- file.path(tempdir(),"HaplotypeEval")
chr    <- unique(map$Chr)
GTfile <- file.path( dir, paste("Chr", chr, ".phased",     sep=""))
files  <- haplofreq(GTfile, Cattle, map, thisBreed="Angler", w.dir=wdir)

fN     <- segN(files$match, map)
mean(fN)
#[1] 0.15418

fN     <- segN(files$match, map, cores=NA)
mean(fN)
#[1] 0.15418

#unlink(wdir, recursive = TRUE)
```

`sim2dis` *Converts a Similarity Matrix into a Dissimilarity Matrix*

Description

Converts a similarity matrix (e.g. a kinship matrix) into a dissimilarity matrix.

Usage

```
sim2dis(f, a=4.0, baseF=0.03, method=1)
```

Arguments

`f`	Similarity matrix.
`a`	Exponent
`baseF`	Old inbreeding not measured by `f`
`method`	Either `1` or `2`

Details

This function converts a similarity matrix `f` with values between 0 and 1 (e.g. a kinship matrix) into a dissimilarity matrix. At first, the similarity is adjusted as

`f <- baseF + (1-baseF)*f`.

Then, for Method 1, the dissimilarity between individuals `i` and `j` is computed as

`Dij = (-log(fij))^a`,

whereas for Method 2, the dissimilarity is computed as

`Dij = sqrt((fii+fjj)/2-fij)^a`.

Although Method 2 may provide lower stress values in some cases, Method 1 has the advantage that the area reflects the diversity of a population more reasonable.

Value

Dissimilarity matrix D.

Author(s)

Robin Wellmann

Examples

```
data(map)
dir   <- system.file("extdata", package = "optiSel")
files <- file.path(dir, paste("Chr", unique(map$Chr), ".phased", sep=""))
f     <- segIBD(files, map, minSNP=15, minL=1.0)
D     <- sim2dis(f, 4)

## Multidimensional scaling of animals:

data(Cattle)
library("smacof")
color <- c(Angler="red", Rotbunt="green", Fleckvieh="blue", Holstein="black")
col   <- color[as.character(Cattle$Breed)]
Res   <- smacofSym(D, itmax = 5000, eps = 1e-08)
plot(Res$conf, pch=18, col=col, main="Multidimensional Scaling", cex=0.5)
mtext(paste("segIBD Stress1 = ", round(Res$stress,3)))
```

---

subPed *Creates a Subset of a Large Pedigree*

Description

Creates a subset of a large pedigree that includes only individuals related with specified individuals in a predefined way.

Usage

```
subPed(Pedig, keep, prevGen=3, succGen=0)
```

Arguments

Pedig	Data frame containing the pedigree where the first 3 columns correspond to: Individual ID, Sire, and Dam. More columns can be passed in the Pedig argument including columns named Sex, Breed (with breed names), and Born (with years of birth). Missing parents are coded as NA, 0, or "0".
keep	Vector with IDs of individuals. Only these individuals and individuals related with them in a predefined way will be kept in the pedigree.
prevGen	Number of previous (ancestral) generations to be included in the pedigree.
succGen	Number of succeeding (descendant) generations to be included in the pedigree.

Details

This function creates a subset of a large pedigree that includes only individuals related with the individuals specified in the vector keep in a predefined way.

Value

A data frame containing the reduced pedigree. A column keep is appended indicating which individuals were included in parameter keep.

Author(s)

Robin Wellmann

Examples

```
data(PedigWithErrors)

sPed <- subPed(PedigWithErrors, keep="276000891974272", prevGen=3, succGen=2)
sPed

label <- c("Indiv", "Born", "Breed")
pedplot(sPed, mar=c(2,4,2,4), label=label, cex=0.7)
```

---

summary.candes *Population Genetic Parameters at Different Times*

Description

For every time point (age cohort), several population genetic parameters are estimated. These may include the generation interval, the average kinship, the average native kinship, the native effective size (native Ne), and the native genome equivalent (NGE) of the population at that point in time.

Usage

```
## S3 method for class 'candes'
summary(object, tlim=range(object$phen$Born, na.rm=TRUE),
     histNe=NA, base=tlim[1], df=4, ...)
```

Arguments

object	R-Object created with function candes containing phenotypes and kinship information on individuals from the same breed. Data frame cand$phen has columns Indiv (with IDs of the individuals), Born (with the years-of-birth or generation-numbers), Sex, I (average age of the parents at date of birth), and Offspring (indicating if the individual has offspring). Typically function prePed is used to create them. For computing the native Ne the individuals must be from different age cohorts.
tlim	Numeric vector with 2 components giving the time span for which genetic parameters are to be computed.
histNe	The historic effective size of the population assumed for the time between year base and tlim[1], which affects the NGE.
base	The base year in which individuals are assumed to be unrelated. The base year affects the NGE. The default is tlim[1].
df	Smoothing parameter used for computing the native effective size. The default is df=4.
...	further arguments passed to or from other methods

Details

For every time point (age cohort), several population genetic parameters are estimated. These may include the generation interval, the average kinship, the average native kinship, the native effective size (native Ne), and the native genome equivalent (NGE) of the population at that point in time. The population at a time t consists of all individuals born between t-I and t, where I is the generation interval. The population genetic parameters are described below.

Value

A data frame providing for each time point (age cohort) several population genetic parameters. These may include

t	The age cohort, containing e.g. year-of-birth or the generation number. These are the levels of column Born from data frame cand$phen.
I	The estimated generation interval at the time when the individuals were born.
KIN	The average kinship KIN in the population at the time when the individuals were born, where KIN is the name of a kinship. It is an estimate of the probability that 2 alleles chosen from the population are IBD.
NATKIN	The average native kinship NATKIN in the population at the time when the individuals were born, where NATKIN is the name of a native kinship. It is an estimate of the conditional probability that 2 alleles chosen from the population are IBD, given that both are from native ancestors.

Ne — The native effective size of the population at the time when the individuals were born. The native effective size, quantifies how fast the smoothed native kinship is increasing. The native kinship may decrease for a short time span, in which case the estimate would be NA. Use a smaller value for parameter df to get a smoother estimate.

NGE — The native genome equivalents of the population at the time when the individuals were born. The NGE estimates the number of unrelated individuals that would be needed to establish a hypothetical new population that has the same genetic diversity at native alleles as the population under study, whereby the individuals born in the base-year are assumed to be unrelated.

Author(s)

Robin Wellmann

Examples

```
data(ExamplePed)
Pedig   <- prePed(ExamplePed, thisBreed="Hinterwaelder", lastNative=1970)
phen    <- Pedig[Pedig$Breed=="Hinterwaelder",]
pKin    <- pedIBD(Pedig)
pKinatN <- pedIBDatN(Pedig, thisBreed="Hinterwaelder")
pop     <- candes(phen=phen, pKin=pKin, pKinatN=pKinatN, quiet=TRUE, reduce.data=FALSE)
Param   <- summary(pop, tlim=c(1970,1995), histNe=150, base=1800, df=4)

plot(Param$t, Param$pKinatN,type="l", ylim=c(0,0.1))
lines(Param$t,Param$pKin, col="red")

plot(Param$t, Param$Ne,  type="l", ylim=c(0,100))
plot(Param$t, Param$NGE, type="l", ylim=c(0,10))
plot(Param$t, Param$I,   type="l", ylim=c(0,10))
```

---

summary.Pedig — *Calculates Summary Statistics for Pedigrees.*

Description

Calculates summary statistics for pedigrees.

Usage

```
## S3 method for class 'Pedig'
summary(object, keep.only=NULL, maxd=50, d=4, ...)
```

Arguments

object	An object from class Pedig, which is usually created with function prePed.
keep.only	The individuals to be included in the summary.
maxd	Maximum pedigree depth.
d	Number of generations taken into account for computing the PCI.
...	further arguments passed to or from other methods

Details

Computes summary statistics for pedigrees, including the numbers of equivalent complete generations, numbers of fully traced generations, numbers of maximum generations traced, indexes of pedigree completeness (MacCluer et al, 1983), and the inbreeding coefficients.

Value

A data frame with the following columns:

Indiv	IDs of the individuals,
equiGen	Number of equivalent complete generations,
fullGen	Number of fully traced generations,
maxGen	Number of maximum generations traced,
PCI	Index of pedigree completeness (MacCluer et al, 1983) in generation d.
Inbreeding	Inbreeding coefficient.

Author(s)

Robin Wellmann

References

MacCluer J W, Boyce A J, Dyke B, Weitkamp L R, Pfenning D W, Parsons C J (1983). Inbreeding and pedigree structure in Standardbred horses. J Hered 74 (6): 394-399.

Examples

```
data(PedigWithErrors)
Pedig <- prePed(PedigWithErrors)
Summary <- summary(Pedig, keep.only=Pedig$Born %in% (2006:2007))
head(Summary)

hist(Summary$PCI,         xlim=c(0,1),  main="Pedigree Completeness")
hist(Summary$Inbreeding, xlim=c(0,1),  main="Inbreeding")
hist(Summary$equiGen,     xlim=c(0,20), main="Number of Equivalent Complete Generations")
hist(Summary$fullGen,     xlim=c(0,20), main="Number of Fully Traced Generations")
hist(Summary$maxGen,      xlim=c(0,20), main="Number of Maximum Generations Traced")
```

Index