Package 'optiSel'

May 26, 2017

Type Package

Title Optimum Contribution Selection and Population Genetics

Version 0.9.1

Date 2017-05-26

Author Robin Wellmann

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

Depends R (>= 3.3.2)

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.

License GPL-2

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

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, ggplot2, rmarkdown

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-05-26 07:26:12 UTC

R topics documented:

`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.

Details

Optimum Contribution Selection

The following functions can be used to compute optimum contributions:

opticont	Calculates optimum genetic contributions of selection candidates to the next generation,
opticont4mb	Calculates optimum genetic contributions of selection candidates to the next generation using multi-breed genotype data,
opticomp	Calculates breed composition of a multi-breed population with maximum diversity,

Function noffspring can then be used to compute the optimum numbers of offspring of selection candidates from their optimum contributions, and function matings can be used for mate allocation. The available constraint settings and objective functions for a particular data set can be determined with the following help functions:

help.opticont	Displays available objective functions and constraints for function opticont,
help.opticont4mb	Displays available objective functions and constraints for function opticont4mb.

For checking the results use

summary.opticont	Calculates genetic parameters of the offspring population, and checks for validity of optimum genetic contributions.

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.

Results from these functions can be combined with function kinlist into a single R object, which can then be used as an argument to function opticont or opticont4mb. 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 contrained or maximized with functions opticont or opticont4mb to remove introgressed genetic material, or alternatively, they 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	Sampling 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,
kinwac	Calculates for every individual it's mean **kin**ship **w**ith the **a**ge **c**ohort to which it belongs,
summary.kinMatrices	Calculates for every age cohort several genetic parameters. These may include average kinships, genetic diversities, gene diversity at native loci, the native effective size, and the native genome equivalent.

Genotype File 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.

Author(s)

Robin Wellmann

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

Examples

```
#See ?opticont for optimum contribution selection
#See ?opticont4mb for multi-breed 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          #
##########################################

Kinships <- kinlist(pedIBD=pedIBD(Pedig), pedIBDatN=pedIBDatN(Pedig, thisBreed="Hinterwaelder"))
Param  <- summary(Kinships, Pedig, tlim=c(1970,2005), histNe=150, base=1800, df=4)

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

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

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

plot(Param$cohort, Param$condGD, type="l", ylim=c(0,1),
     main="Diversity at native alleles",ylab="condGD",xlab="")
mtext(info, cex=0.7)

plot(Param$cohort,Param$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")
```

`Cattle` *Phenotypes of Genotyped Cattle*

Description

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 simulated breeding value (`BV`), a simulated 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')	Level of the grouping factor,
Generation	Generation number,
Completeness	Completeness of the pedigree.

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)

## Not run:
library("ggplot2")
ggplot(compl, aes(Generation, Completeness, col=Sex))+geom_line()

## End(Not run)
```

---

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

```
ExamplePed
```

Format

A data frame with columns for individual ID, sire ID, dam ID, sex, breed, year of birth, and a simulated breeding value

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)
```

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.

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)



## Creating output files with allele frequencies and allele origins:
## Not run:
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)

## End(Not run)
```

`help.opticont` *Displays Available Objective Functions and Constraints for Function* `opticont`.

Description

Displays available objective functions and constraints for function opticont.

Usage

```
help.opticont(K, phen)
```

Arguments

K	Named list containing one or more NxN Kinship matrices for the N breeding animals.
phen	Data frame with N rows and various columns, with animal IDs in column 1 and sex in column 2. The other columns may contain breeding values or migrant contributions.

Details

Lists available objective functions and constraints for function opticont.

Author(s)

Robin Wellmann

Examples

```
#see ?opticont
```

help.opticont4mb — *Displays Available Objective Functions and Constraints for Function* opticont4mb.

Description

Displays available objective functions and constraints for function opticont4mb.

Usage

```
help.opticont4mb(K, phen)
```

Arguments

K	Named list containing one or more kinship matrices.
phen	Data frame with N rows and various columns, with animal IDs in column 1 and sex in column 2. The other columns may contain breeding values or migrant contributions.

Details

Lists available objective functions and constraints for function opticont4mb.

Author(s)

Robin Wellmann

Examples

```
#see ?opticont4mb
```

Kin — *A list of class kinMatrices containing kinship matrices*

Description

A list of class kinMatrices containing kinship matrix fA and the matrices and the matrices needed for computing fD for the animals included in datat set Phen.

Usage

```
Kin
```

Format

A list of class kinMatrices.

Examples

```
data(PedigWithErrors)
data(Phen)
Phen[1:5,]

keep  <- Phen$Indiv
Pedig <- prePed(PedigWithErrors, keep=keep, thisBreed="Hinterwaelder", lastNative=1970)
fA    <- pedIBD(Pedig, keep.only=keep)
fD    <- pedIBDatN(Pedig, thisBreed="Hinterwaelder",  keep.only=keep)
Kin2   <- kinlist(fA=fA, fD=fD)

data(Kin)

identical(Kin, Kin2)
#[1] TRUE
```

---

kinlist *Combines Kinship Matrices into a List*

Description

Combines matrices or objects of class kinMatrices into a list

Usage

```
kinlist(...)
```

Arguments

...	One or more matrices or objects of class kinMatrices

Details

Combines matrices or objects of class kinMatrices into a list

Value

List of class kinMatrices

Author(s)

Robin Wellmann

Examples

```
data(PedigWithErrors)
data(Phen)
Phen  <- Phen[80:140,]
keep  <- Phen$Indiv
Pedig <- prePed(PedigWithErrors, keep=keep, thisBreed="Hinterwaelder", lastNative=1970)
fA    <- pedIBD(Pedig, keep.only=keep)
fD    <- pedIBDatN(Pedig, thisBreed="Hinterwaelder",  keep.only=keep)
x     <- pedIBDorM(Pedig, thisBreed="Hinterwaelder", keep.only=keep)
Kin   <- kinlist(fA=fA, fB=x$pedIBDorM, fC=x$pedIBDorMM, fD=fD)
help.opticont(Kin, Phen)
```

---

kinwac *Calculates Kinships With Age Cohorts*

Description

Calculates for every individual it's mean kinship with the age cohort to which it belongs (excluding the kinship with itself).

Usage

```
kinwac(K, Pedig)
```

Arguments

K	Named list containing kinship matrices of the individuals to be used for computing mean kinships. Typically only individuals from the breed of interest are included.
Pedig	Data frame containing the Pedigree. The data frame has columns (1) Individual, (2) Sire, (3) Dam, and column Born containing for every individual the age cohort to which it belongs (typically year of birth).

Details

The function computes for every individual it's mean kinship with the age cohort to which it belongs. This can be computed for different notions of kinship simultaneously. If an individual has high kinship with it's age cohort then it should not be extensively used for breeding if the aim is to conserve genetic diversity.

Value

A data frame with column names `c("cohort", "I", names(K))` containing for every individual and every matrix provided in K the average value of the individual with the birth cohort to which it belongs.

The data frame has class attribute "kinwac".

Author(s)

Robin Wellmann

Examples

```
data(ExamplePed)
Pedig    <- prePed(ExamplePed, thisBreed="Hinterwaelder", lastNative=1970)
Kinships <- kinlist(pedIBD=pedIBD(Pedig), pedIBDatN=pedIBDatN(Pedig, thisBreed="Hinterwaelder"))
Kin      <- kinwac(Kinships, Pedig=Pedig)
Kin[1001:1010,]
```

---

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 offspring. The mean inbreeding coefficient in the offspring is minimized.

Usage

```
matings(cand, Kin, N=2*sum(cand$Sex=="female"), alpha=1, ub.nOff=NA, max=FALSE, ...)
```

Arguments

cand	Data frame with optimum contributions (column oc), 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.
N	Desired number of offspring that should be available as selection candidates in the next generation.
alpha	If alpha<1 then the proportion of females mated 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.nOff	Maximum number of offspring per mating. Without this constraint (i.e. ub.nOff=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.
...	Further optimization parameters passed to function ecos.control.

Details

Males and females are allocated for mating such that all breeding animals have the desired number of offspring. If Kin is a kinship matrix, then the mean inbreeding coefficient in the offspring is minimized. The maximum number of offspring per mating can be constrained. For each herd, the proportion alpha of females mated with the same male can be constrained as well, but this increases computation time.

Note that the desired number of offspring per mating is the number of offspring that should be used as selection candidates in the next generation, which is not necessarily the total number of offspring.

Value

Data frame with columns Sire, Dam, nOff, and herd, whereby column nOff contains the desired number of offspring from each mating, and column herd contains the herd of the dam.

Author(s)

Robin Wellmann

Examples

```
data(map)
dir   <- system.file("extdata", package = "optiSel")
files <- paste(dir, "/Chr", 1:2, ".phased", sep="")
sKin  <- segIBD(files, map, minSNP=20, minL=2.0)
Kin   <- kinlist(sKin = sKin)

data(Cattle)
Phen  <- Cattle[Cattle$Breed=="Angler", ]
```

```
head(Phen)

help.opticont(Kin, Phen)

con   <- list(ub.sKin = 0.057)
maxBV <- opticont("max.BV", K=Kin, phen=Phen, con=con, solver="cccp2", trace=FALSE)
(summary(maxBV))

#####  Minimize inbreeding   #####
Candidate <- maxBV$parent
Mating <- matings(Candidate, sKin, ub.nOff=5, maxit=50L)
Mating
attributes(Mating)$objval
```

`noffspring` *Calculates Optimum Numbers of Offspring*

Description

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

Usage

```
noffspring(cand, N)
```

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.

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 integer column `nOff` containing the optimum numbers of offspring. This column is approximately `2*N*cand$oc` with `sum(noff[cand$Sex=="male"])=N` and `sum(noff[cand$Sex=="female"])=N`.

Author(s)

Robin Wellmann

Examples

```
data(PedigWithErrors)
data(Phen)

keep  <- Phen$Indiv
Pedig <- prePed(PedigWithErrors, keep=keep, thisBreed="Hinterwaelder", lastNative=1970)
Kin   <- kinlist(fA=pedIBD(Pedig, keep.only=keep))
con   <- list(ub.fA=0.03, ub=c(M=NA, F=-1))
maxBV <- opticont("max.BV", K=Kin, phen=Phen, con=con, trace = FALSE)
summary(maxBV)

N <- 150
cand <- maxBV$parent
cand$nOff <- noffspring(cand, N)$nOff

sum(cand$nOff[cand$Sex=="male"])
#[1] 150

sum(cand$nOff[cand$Sex=="female"])
#[1] 150

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

---

opticomp *Calculates the Optimum Breed Composition*

Description

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

Usage

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

Arguments

f	Kinship matrix (e.g. a segment based kinship matrix, or a IBS based kinship matrix).
Breed	Vector containing the breed name of each individual.
obj.fun	The objective function to be maximized. For "NGD" the diversity 1-**c**'**fc** is maximized, or equivalently, the mean kinship **c**'**fc** is minimized, 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 synthetic 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 synthetic 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=15, map=map, minL=1.0)
mb  <- opticomp(IBD, Breed=Cattle$Breed, obj.fun="NGD")
```

```
#### Optimum breed composition: ###
round(mb$bc,3)
#   Angler Fleckvieh  Holstein   Rotbunt
#    0.484     0.395     0.072     0.049

#### Average kinships within and between breeds: ###
round(mb$f,4)
#          Angler Fleckvieh Holstein Rotbunt
#Angler    0.0699    0.0171   0.0584  0.0590
#Fleckvieh 0.0171    0.0956   0.0135  0.0138
#Holstein  0.0584    0.0135   0.1233  0.1065
#Rotbunt   0.0590    0.0138   0.1065  0.1227

#### Genetic distances between breeds: ###
round(mb$Dist,4)
#          Angler Fleckvieh Holstein Rotbunt
#Angler    0.0000    0.2561   0.1953  0.1930
#Fleckvieh 0.2561    0.0000   0.3098  0.3087
#Holstein  0.1953    0.3098   0.0000  0.1284
#Rotbunt   0.1930    0.3087   0.1284  0.0000

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

bc <- opticomp(IBD, Breed=Cattle$Breed, obj.fun="NTD")$bc
round(bc,3)
#   Angler Fleckvieh  Holstein   Rotbunt
#    0.253     0.433     0.168     0.146
```

`opticont` *Calculates Optimum Contributions of Selection Candidates*

Description

Calculates optimum genetic contributions of selection candidates to the next generation.

Usage

```
opticont(method, K, phen, con=list(), solver="auto", quiet=FALSE,
    make.definite=solver=="csdp", ...)
```

Arguments

`method`	Possible values are `"min.VAR"`, and `"max.VAR"`, where `VAR` is the name of a column in data frame `phen`, or `"min.KIN"`, where `KIN` is the name of a kinship as defined by function kinlist. Use help.opticont to see the available objective functions.

K	List created by function kinlist, containing e.g. kinship matrices of the selection candidates.
phen	Data frame with IDs of selection candidates in column 1, and sex in column 2. The other columns may contain breeding values or migrant contributions. The sex is coded as 'male' and 'female'.
con	List defining the constraints. The compomnents are described in the Details section. If a component is missing, then the respective constraint is not applied. Use help.opticont to see the available constraints.
solver	Name of the algorithm for optimization. Available solvers are "alabama", "cccp", "cccp2", "csdp", and "slsqp". The default "auto" means that the solver is chosen automatically. The solvers are described in the Details section.
quiet	If quiet=FALSE then detailed information is shown.
make.definite	If make.definite=TRUE then all non-definite matrices are approximated by positive definite matrices before optimization. This is the default setting for the solver csdp.
...	Tuning parameters of the solver. The available parameters depend on the solver and will be printed when function opticont is used with default values. An overview is given in the Details section.

Details

Computation of optimum genetic contributions of selection candidates.

Constraints

Constraints are defined in argument con, which is a list whose components may have the following names:

ub.KIN: Upper bound for the mean kinship in the offspring, where KIN must be replaced by the name of a kinship as defined by function kinlist. Upper bounds for an arbitrary number of different kinships may be provided.

lb: Either a named vecor of the form c(M=a, F=b) containing lower bounds for the contributions of males (a) and females (b), or a N-vector containing the minimum permissible contribution of each selection candidate. The default is c(M=0, F=0).

ub: Either a named vecor of the form c(M=a, F=b) containing upper bounds for the contributions of males (a) and females (b), or a N-vector containing the maximum permissible contribution of each selection candidate. For M=-1 (F=-1) it is assumed that all males (females) have equal contributions to the offspring. If a number is NA then the number of offspring for that sex/individual is not bounded. The default is c(M=NA, F=NA).

lb.VAR: Lower bound for the expected mean value of variable VAR from data frame phen in the offspring. For example lb.BV=a defines a lower bound for the mean breeding value in the offspring to be a if data frame phen has column BV with breeding values of the parents. Lower bounds for an arbitrary number of variables can be defined.

ub.VAR: Upper bound for the mean value of variable VAR from data frame phen in the offspring. For example ub.MC=a defines the upper bound for the genetic contributions from migrant breeds in the offspring to be a if data frame phen has column MC with migrant contributions for the parents. Upper bounds for an arbitrary number of variables can be defined.

eq.VAR: Equality constraint for the mean value of variable VAR from data frame phen in the offspring. Equality constraints for an arbitrary number of variables can be defined.

Solver

"alabama": The augmented lagrangian minimization algorithm auglag from package alabama. The method combines the objective function and a penalty for each constraint into a single function. This modified objective function is then passed to another optimization algorithm with no constraints. If the constraints are violated by the solution of this sub-problem, then the size of the penalties is increased and the process is repeated. The default methods for the uncontrained optimization in the inner loop is the quasi-Newton method called BFGS. The available parameters used for the outer loop are described in the details section of the help page of function auglag. The available parameters used for the inner loop are described in the details section of the help page of function optim.

"cccp", "cccp2": Function cccp from package cccp for solving cone constrained convex programs. For cccp quadratic constraints are defined as second order cone constraints, whereas for cccp2 quadratic constraints are defined by functions. The implemented algorithms are partially ported from CVXOPT. The parameters are those from function ctrl. They are among others the maximum count of iterations as an integer value (maxiters), the feasible level of convergence to be achieved (feastol) and whether the solver's progress during the iterations is shown (trace). If numerical problems are encountered increase the optimization parameter feastol or reduce parameter stepadj.

"csdp": The problem is reformulated as a semidefinite programming problem and solved with the CSDP library. Non-definite matrices are approximated by positive definite matrices. This solver is not suitable when the objective is to minimize kinship at native alleles. Available parameters are described in the CSDP User's Guide: https://projects.coin-or.org/Csdp/export/49/trunk/doc/csdpuser.pdf.

"slsqp": The sequential (least-squares) quadratic programming (SQP) algorithm slsqp for gradient-based optimization from package nloptr. The algorithm optimizes successive second-order (quadratic/least-squares) approximations of the objective function, with first-order (affine) approximations of the constraints. Available parameters are described in nl.opts

Remark

If the function does not provide a valid result due to numerical problems then try the following modifications:

* modify the optimization parameters,
* use another solver,
* change the order of the kinship constraints if more than one kinship is constrained,
* define upper or lower bounds instead of equality constraints.
* increase the upper bounds for the kinships.

Validity of the result can be checked with function summary.opticont. Use help.opticont to see available objective functions and constraints.

Value

A list with class "opticont" which has component parent. This is the data frame phen with the additional column oc containing the optimum genetic contribution of each selection candidate to

the next generation, lb containing the lower bounds, and ub containing the upper bounds.

Author(s)

Robin Wellmann

References

Borchers, B. (1999). CSDP, A C Library for Semidefinite Programming Optimization Methods and Software 11(1):613-623 http://euler.nmt.edu/~brian/csdppaper.pdf

Kraft, D. (1988). A software package for sequential quadratic programming, Technical Report DFVLR-FB 88-28, Institut fuer Dynamik der Flugsysteme, Oberpfaffenhofen, July 1988.

Lange K, Optimization, 2004, Springer.

Madsen K, Nielsen HB, Tingleff O, Optimization With Constraints, 2004, IMM, Technical University of Denmark.

Examples

```
################################################
# Example 1: Advanced OCS using pedigree data   #
# Objective: maximize genetic gain              #
# Constraints:                                  #
#   - mean kinship                              #
#   - mean kinship at native alleles            #
#   - genetic contributions from other breeds   #
################################################
data(PedigWithErrors)
data(Phen)

keep  <- Phen$Indiv
Pedig <- prePed(PedigWithErrors, keep=keep, thisBreed="Hinterwaelder", lastNative=1970)
Pedig$MC <- 1-pedBreedComp(Pedig, thisBreed="Hinterwaelder")$native
Phen  <- merge(Pedig, Phen[,c("Indiv", "BV")], by="Indiv")
Kin   <- kinlist(pKin    = pedIBD(Pedig, keep.only=keep),
                 pKinatN = pedIBDatN(Pedig, thisBreed="Hinterwaelder",  keep.only=keep))

head(Phen)
cor(Phen$MC, Phen$BV)
help.opticont(Kin, Phen)

# Compute offspring parameters for unselected population
noSel      <- opticont(method="min.pKin", K=Kin, phen=Phen, con=list(ub=c(M=-1, F=-1)))
noSel.s    <- summary(noSel)
round(noSel.s[,c("pKin","pKinatN", "meanMC", "meanBV")],4)
meanMC     <- noSel.s$meanMC
meanKin    <- noSel.s$pKin
meanKinatN <- noSel.s$pKinatN
meanBV     <- noSel.s$meanBV

# Define Constraints
Ne <- 100
```

```
con            <- list(ub=c(M=NA, F=-1))
con$ub.pKin    <- meanKin    + (1-meanKin   )*(1/(2*Ne))
con$ub.pKinatN <- meanKinatN + (1-meanKinatN)*(1/(2*Ne))
con$ub.MC      <- 0.97*meanMC

# Compute the genetic progress achievable
maxBV   <- opticont("max.BV", K=Kin, phen=Phen, con=con)
maxBV.s <- summary(maxBV)
maxBV.s$meanBV
# [1] 0.5428925

# Get optimum contributions of sires
Sire <- maxBV$parent[maxBV$parent$Sex=="male",]
ord  <- order(Sire$oc, decreasing=TRUE)
head(Sire[ord,])

################################################
# Example 2: Advanced OCS using genotype data  #
# Objective: minimize inbreeding               #
# Constraints:                                 #
#   - breeding values                          #
#   - mean kinship at native alleles           #
#   - genetic contributions from other breeds  #
################################################

data(map)
data(Cattle)
dir   <- system.file("extdata", package = "optiSel")
files <- paste(dir, "/Chr", 1:2, ".phased", sep="")
Kin   <- kinlist(sKin    = segIBD(files, map, minSNP=20, minL=2.0),
                 sKinatN = segIBDatN(files, Cattle, map, thisBreed="Angler",
                                     ubFreq=0.01, minSNP=20, minL=2.0))
Haplo <- haplofreq(files, Cattle, map, thisBreed="Angler",
                   minSNP=20, minL=2.0, ubFreq=0.01, what="match")
Comp  <- segBreedComp(Haplo$match, map)
Comp$MC <- 1-Comp$native
Phen  <- merge(Cattle, Comp[,c("Indiv", "MC")], by="Indiv", all=FALSE)

help.opticont(Kin, Phen)

cor(Phen$MC, Phen$BV,use="complete.obs")
#[1] 0.5033714

# Compute offspring parameters for unselected population
noSel      <- opticont(method="min.sKin", K=Kin, phen=Phen, con=list(ub=c(M=-1, F=-1)))
noSel.s    <- summary(noSel)
round(noSel.s[,c("sKin","sKinatN", "meanMC", "meanBV")],4)
meanMC     <- noSel.s$meanMC
meanKin    <- noSel.s$sKin
meanKinatN <- noSel.s$sKinatN
meanBV     <- noSel.s$meanBV
```

```
# Define Constraints
Ne <- 100

con            <- list(ub=c(M=NA, F=-1))
con$ub.sKinatN <- meanKinatN + (1-meanKinatN)*(1/(2*Ne))
con$ub.MC      <- 0.97*meanMC
con$lb.BV      <- meanBV

# Compute the smallest mean kinship achievable
minKin   <- opticont("min.sKin", K=Kin, phen=Phen, con=con)
minKin.s <- summary(minKin)
minKin.s$sKin
# [1] 0.03881304

# Get optimum contributions of sires
Sire <- minKin$parent[minKin$parent$Sex=="male",]
ord  <- order(Sire$oc, decreasing=TRUE)
head(Sire[ord,])
```

`opticont4mb` *Calculates Optimum Contributions of Selection Candidates using Multi-Breed Genotype Data*

Description

Calculates optimum genetic contributions for selection candidates from one breed using multi-breed genotype data. Genotype data from multiple breeds may be used in order to increase the genetic distance between the breed of interest (`thisBreed`) and other breeds.

Usage

```
opticont4mb(method, K, phen, bc, thisBreed=names(bc)[1], con=list(),
    solver="cccp", quiet=FALSE, make.definite=solver=="csdp", ...)
```

Arguments

`method`	Possible values are `"min.VAR"`, and `"max.VAR"`, where `VAR` is the name of a column in data frame `phen`, or `"min.KIN"`, or `"min.KIN.acrossBreeds"`, where `KIN` is the name of a kinship as defined by function kinlist. Use help.opticont4mb to see the available objective functions. If kinship `KIN` is available for all animals from the multi-breed population, then `"min.KIN.acrossBreeds"` minimizes the kinship in the multi-breed population by optimizing the contributions of selection candidates from the breed of interest.
`K`	List created by function kinlist containing kinships of genotyped individuals.
`phen`	Data frame with one row for each animal from the multi-breed population that is to be included in the analysis. The animal IDs is in column 1 (named `Indiv`) and the sex is in column 2 (named `Sex`). The sex is coded as `'male'` and `'female'`. Column `Breed` contains the breed name of every genotyped animal. Further

columns contain e.g. breeding values or migrant contributions that may be used for defining linear constraints.

`bc` Named vector containing the proportion of every genotyped breed in the hypothetical multi-breed offspring population. The names of the components are the breed names. Note that only the contributions of selection candidates from `thisBreed` will be optimized. Animals from other breeds have fixed contributions.

`thisBreed` The breed to which the selection candidates belong.

`con` List defining the constraints. The components are described in the Details section. If a component is missing, then the respective constraint is not applied. Use help.opticont4mb to see the available constraints.

`solver` Name of the algorithm for optimization. Available solvers are `"alabama"`, `"cccp"`, `"cccp2"`, `"csdp"`, and `"slsqp"`. The default is `"cccp"`. The solvers are described in the Details section.

`quiet` If `quiet=FALSE` then detailed information is shown.

`make.definite` If `make.definite=TRUE` then all non-definite matrices are approximated by positive definite matrices before optimization. This is the default setting for the solver `csdp`.

`...` Tuning parameters of the solver. The available parameters depend on the solver and will be printed when function `opticont` is used with default values. An overview is given in the Details section.

Details

Computation of optimum genetic contributions for the selection candidates from one breed using multi-breed marker data. Marker data from multiple breeds may be used in order to increase the genetic distance between the breed of interest (`thisBreed`) and the other breeds.

In this case a hypothetical subdivided population is considered consisting of purebred offspring of genotyped individuals. That is, the offspring population consists of several breeds with specified breed proportions (e.g. 20% Angler cattle, 40% Holstein cattle, and 40% Fleckvieh cattle). Only the contributions of the selection candidates from `thisBreed` will be optimized. Animals from other breeds have equal contributions.

The aim is to reduce the average genomic relationship in this multi-breed population since this causes the genetic distance between `thisBreed` and other breeds to increase. This may increase the conservation value of the breed.

If managing diversity across breeds is not intended then function opticont could be used instead.

Constraints

A list possibly containing the following components providing the constraints:

ub.KIN: Upper bound for the mean kinship in the offspring, where `KIN` must be replaced by the name of a kinship as defined by function kinlist. Use help.opticont4mb to see available methods.

ub.KIN.acrossBreeds: Upper bound for the mean kinship in the next generation of the multi-breed population, where `KIN` must be replaced by the name of a kinship as defined by function kinlist. Use help.opticont4mb to see available methods.

lb: Either a named vecor of the form `c(M=a, F=b)` containing lower bounds for the contributions of males (`a`) and females (`b`) from `thisBreed`, or a named vector containing the minimum permissible contribution of each selection candidate. The default is `c(M=0, F=0)`.

ub: Either a named vecor of the form `c(M=a, F=b)` containing upper bounds for the contributions of males (`a`) and females (`b`) from `thisBreed`, or a named vector containing the maximum permissible contribution of each selection candidate. For `M=-1` (`F=-1`) it is assumed that all males (females) have equal contributions to the offspring. If a number is `NA` then the number of offspring for that sex/individual is not bounded. The default is `c(M=NA, F=NA)`.

lb.VAR: Lower bound for the mean value of variable `VAR` from data frame `phen` in the offspring from `thisBreed`. For example `lb.BV=a` defines a lower bound for the mean breeding value in the offspring from `thisBreed` to be `a` if data frame `phen` has column `BV` with breeding values of the parents. Lower bounds for an arbitrary number of variables can be defined.

ub.VAR: Upper bound for the mean value of variable `VAR` from data frame `phen` in the offspring from `thisBreed`. For example `ub.MC=a` defines the upper bound for the genetic contributions from migrant breeds in the offspring of `thisBreed` to be `a` if data frame `phen` has column `MC` with migrant contributions for the parents. Upper bounds for an arbitrary number of variables can be defined.

eq.VAR: Equality constraint for the mean value of variable `VAR` from data frame `phen` in the offspring from `thisBreed`. For example `eq.MC=a` forces the genetic contribution from migrant breeds in the offspring from `thisBreed` to be `a` if data frame `phen` has column `MC` with migrant contributions for the parents. Equality constraints for an arbitrary number of variables can be defined.

Solver

`"alabama"`: The augmented lagrangian minimization algorithm auglag from package `alabama` is used. That is, the method combines the objective function and a penalty for each constraint into a single function. This modified objective function is then passed to another optimization algorithm with no constraints. If the constraints are violated by the solution of this sub-problem, then the size of the penalties is increased and the process is repeated. The default methods for the uncontrained optimization in the inner loop is the quasi-Newton method called `BFGS`. The available parameters used for the outer loop are described in the details section of the help page of function auglag. The available parameters used for the inner loop are described in the details section of the help page of function optim.

`"cccp"`, `"cccp2"`: Function cccp from package `cccp` for solving cone constrained convex programs is used. For `cccp` quadratic constraints are defined as second order cone constraints. This solver is not suitable if computation of the Cholesky decomposition fails. For `cccp2` quadratic constraints are defined by functions. The implemented algorithms are partially ported from CVXOPT. The parameters are those from function ctrl. They are among others the maximum count of iterations as an integer value (`maxiters`), the feasible level of convergence to be achieved (`feastol`) and whether the solver's progress during the iterations is shown (`trace`). If numerical problems are encountered increase the optimization parameter `feastol` or reduce parameter `stepadj`.

`"csdp"`: The problem is reformulated as a semidefinite programming problem and solved with the CSDP library. Non-definite matrices are approximated by positive definite matrices. This solver is not suitable when the objective is to minimize kinship at native alleles. Available parameters are described in the CSDP User's Guide: `https://projects.coin-or.org/Csdp/export/49/trunk/doc/csdpuser.pdf` .

`"slsqp"`: The sequential (least-squares) quadratic programming (SQP) algorithm slsqp for gradient-based optimization from package `nloptr` is used. The algorithm optimizes successive second-order

(quadratic/least-squares) approximations of the objective function, with first-order (affine) approximations of the constraints. Available parameters are described in nl.opts.

Remark

If the function does not provide a valid result due to numerical problems then try the following modifications:

* modify the optimization parameters,
* use another `solver`,
* change the order of the kinship constraints if more than one kinship is constrained,
* define upper or lower bounds instead of equality constraints.
* increase the upper bounds for the kinships.

Validity of the result can be checked with function summary.opticont. Use help.opticont4mb to see available objective functions and constraints.

Value

A list with class `"opticont"` which has component `parent`. This is the data frame `phen` but includes ony the animals from the breed of interest. It has the additional column `oc` containing the optimum genetic contribution of each selection candidate to the next generation, `lb` containing the lower bounds of the optimum contributions, and `ub` containing the upper bounds.

Author(s)

Robin Wellmann

References

Borchers, B. (1999). CSDP, A C Library for Semidefinite Programming Optimization Methods and Software 11(1):613-623 `http://euler.nmt.edu/~brian/csdppaper.pdf`

Kraft, D. (1988). A software package for sequential quadratic programming, Technical Report DFVLR-FB 88-28, Institut fuer Dynamik der Flugsysteme, Oberpfaffenhofen, July 1988.

Lange K, Optimization, 2004, Springer.

Madsen K, Nielsen HB, Tingleff O, Optimization With Constraints, 2004, IMM, Technical University of Denmark.

Examples

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

### Compute genomic kinship and genomic kinship at native segments
G    <- segIBD(files, map, minSNP=20, minL=3.0)
GN   <- segIBDatN(files, Cattle, map, thisBreed="Angler", refBreeds="others",
           ubFreq=0.02, minSNP=20, minL=3.0, lowMem=TRUE)
```

```
Kin  <- kinlist(G=G, GN=GN)

### Compute migrant contributions of selection candidates
Haplo<- haplofreq(files, Cattle, map, thisBreed="Angler", refBreeds="others",
          minSNP=20, minL=3.0, ubFreq=0.02, what="match")
Comp <- segBreedComp(Haplo$match, map)
Cattle$MC <- NA
Cattle[rownames(Comp), "MC"] <- 1-Comp$native
apply(Comp[,-1],2,mean)
#     native           F           H           R
#0.551844104 0.009739393 0.202216271 0.236200232


########################################
#  Find optimum breed contributions    #
########################################
lb <- c(Angler=0.10, Holstein=0.20, Fleckvieh=0.20)
bc <- opticomp(G, Breed=Cattle$Breed, obj.fun="NGD", lb=lb)$bc
round(bc,3)
#   Angler Fleckvieh  Holstein   Rotbunt
#    0.355     0.445     0.200     0.000

########################################
#  Check available objective functions #
#  and constraints                     #
########################################

help.opticont4mb(Kin, Cattle)


#####################################################################
#   Compute the minimum segment based kinship achievable         #
#  across breeds while constraining it within the breed          #
#####################################################################

con  <- list(ub.G=0.05, ub=c(M=NA, F=-1))
minG <- opticont4mb("min.G.acrossBreeds", Kin, Cattle, bc, thisBreed="Angler", con=con, trace=FALSE)
minG.s <- summary(minG)
minG.s[,c("G.acrossBreeds")]
#[1] 0.02289039

#####################################################################
# Compute the genetic progress achievable while constraining     #
#      segment based kinship  within and across breeds           #
#                     and migrant contributions                  #
#####################################################################

con          <- list(ub.G=0.05, ub.G.acrossBreeds=0.026, ub.MC=0.32,  ub=c(M=NA, F=-1))
maxBV.G.MC   <- opticont4mb("max.BV", Kin, Cattle, bc, thisBreed="Angler", con=con, trace=FALSE)
maxBV.G.MC.s <- summary(maxBV.G.MC)
maxBV.G.MC.s$meanBV
# [1] 0.2851194
```

```
#####################################################################
#    Compute the minimum achievable kinship at native alleles       #
#    while constraining kinship within and across breeds            #
#    and migrant contributions                                      #
#####################################################################

con   <- list(ub.G=0.05, ub.G.acrossBreeds=0.026, ub.MC=0.32, ub=c(M=NA, F=-1))
minGN <- opticont4mb("min.GN", Kin, Cattle, bc, thisBreed="Angler", con=con, solver="slsqp")
minGN.s <- summary(minGN)
minGN.s$GN
#[1] 0.04114953


#####################################################################
# Summary statistics from different optimizations                   #
# can be combined in a data frame. The most important parameters    #
# are printed for comparison:                                       #
#####################################################################
Res <- rbind(minG.s, maxBV.G.MC.s, minGN.s)
format(Res[,c("valid","meanBV", "meanMC", "G.acrossBreeds", "G", "GN")],digits=4)

#           valid  meanBV meanMC G.acrossBreeds       G      GN
#minG        TRUE -0.4329 0.3379        0.02289 0.03451 0.03986
#maxBV.G.MC  TRUE  0.2851 0.3200        0.02475 0.05000 0.06830
#minGN       TRUE -0.5321 0.3200        0.02312 0.03600 0.04115

cor(cbind(minG$parent$oc, maxBV.G.MC$parent$oc, minGN$parent$oc))
#          [,1]      [,2]      [,3]
#[1,] 1.0000000 0.2741736 0.8540842
#[2,] 0.2741736 1.0000000 0.3608900
#[3,] 0.8540842 0.3608900 1.0000000
```

---

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 where the first 3 columns correspond to: 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. If this is not the case use prePed.
thisBreed	Name of this breed as denoted in column (5) 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")
Pedig$MC <- 1-cont$native
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. 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.
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 **ped**igree based probability of alleles to be **IBD at N**ative alleles.

Usage

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

Arguments

Pedig	Data frame containing the pedigree where the first 3 columns correspond to: Individual ID, Sire, and Dam. Additional columns include column Breed with breed names. Missing parents are coded as NA, 0, or "0".
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.
nGen	Number of generations taken into account for estimating the native effective size. The default means that the effective native effective size is not estimated.

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.

Value

A list with the following components:

pedZ	matrix with pedZij = Bij = Probability that two alleles chosen from individuals i and j are IBD or at least one of them is a migrant allele.
pedN	matrix with pedNij = Probability that two alleles chosen from individuals i and j are both native alleles.

The list has class attribute "kinMatrices" and the additional attribute condProb.

Author(s)

Robin Wellmann

Examples

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

#Number of Migrant Founders: 237
#Number of Native  Founders: 150
#Individuals in Pedigree   : 1658
#Native Ne = 49.5 (estimated from 6 previous generations)
#Mean kinship of native alleles:  0.0777
```

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`)

The list has class attribute `"kinMatrices"`.

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.

Usage

```
PedigWithErrors
```

Format

A data frame with columns for individual ID, sire ID, dam ID, sex, breed, and year of birth

---

pedInbreeding *Calculates Pedigree Based Inbreeding*

Description

Calculates Pedigree Based Inbreeding

Usage

```
pedInbreeding(Pedig)
```

Arguments

Pedig	Data frame containing the Pedigree. The data frame has columns (1) Individual, (2) Sire, (3) Dam. Missing parents are coded as NA. Consider preparing the pedigree with function prePed before.

Details

Computation of pedigree based inbreeding. This function is a wrapper function for pedigree from package pedigree and is compatible with the data.table package.

Value

A data frame or data table 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 where the first 3 columns correspond to: Individual ID, Sire, and Dam. 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

```
Phen
```

Format

A data frame with columns for individual ID (Indiv), sex, and breeding values (BV).

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).

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, I=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, 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.
I	offset

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

A data frame containing the pedigree.

Author(s)

Robin Wellmann

Examples

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

---

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 from this breed 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 from this breed 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 from this breed 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 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 from this breed. 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:
## Not run:
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)

## End(Not run)
```

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)
```

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.

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 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)
## Not run:
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)))

## End(Not run)
```

`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)
```

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.

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:
## Not run:
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)
```

```
## End(Not run)
```

---

segIBDatN *Calculates Segment Based Kinship at Native Alleles.*

Description

Segment based probability of alleles to be **IBD at N**ative haplotype segments ("kinship at native segments").

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)
```

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 two 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. 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") and the breed name (column "Breed") 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.

Details

Computation of the segment based probability of alleles to be IBD at native haplotype segments.

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 containing matrices needed for computing the segment based pobability of alleles to be IBD at native segments. The list has components

segN	This matrix contains for each pair of individuals the probability that two SNPs taken at random position from randomly chosen haplotypes both belong to native segments.
segIBDandN	This matrix contains for each pair of individuals the probability that two SNPs taken at random position from randomly chosen haplotypes belong to a shared native segment.
segZ	1+segIBDandN-segN.

The list has attribute `meanIBDatN` providing the probability of randomly chosen alleles to be IBD at native haplotype segments. Note that `1-meanIBDatN` is the genetic diversity at native segments within the genotyped 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="")
Res   <- segIBDatN(files, Cattle, map, thisBreed="Angler", ubFreq=0.01,
                   minL=1.0, lowMem=FALSE)

## Mean kinship at native segments:
attributes(Res)$meanIBDatN
#[1] 0.06695171

## Results for individuals:
kin <-Res$segIBDandN/Res$segN
use <-upper.tri(kin) & Res$segN>0.2
boxplot(kin[use], ylim=c(0,1))

## Use temporary files to reduce working memory:

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

## Mean kinship at native segments:
attributes(Res)$meanIBDatN
#[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)
```

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.

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 or data table 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)
```

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.

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:
## Not run:
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)

## End(Not run)
```

`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.

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:
## Not run:
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)))

## End(Not run)
```

---

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 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.kinMatrices *Calculates Genetic Parameters for Age Cohorts*

Description

Computes for every age cohort several genetic parameters. These are average kinships, gene diversity at native loci, native effective size, and native genome equivalents.

Usage

```
## S3 method for class 'kinMatrices'
summary(object, Pedig, tlim=NULL, histNe=NULL, base=NULL, df=4, ...)
```

Arguments

object	This is typically the output of function kinlist. A named list containing kinship matrices of the individuals to be used for computing mean kinships. Typically individuals from different age cohorts are included, but only from the breed of interest.
Pedig	Data frame containing the Pedigree. The data frame has columns (1) Individual, (2) Sire, (3) Dam, and column Born containing for every individual the age cohort to which it belongs (typically year of birth).
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 before tlim[1]. The default is 3 times the average native effective size within the time span of interest. This parameter is needed only for computing the conditional gene diversity and the native genome equivalents.

base	The base year in which individuals are assumed to be unrelated. The default is 25 generations before tlim[1]. This parameter is needed only for computing the conditional gene diversity and the native genome equivalents.
df	Smoothing parameter used for computing the native effective size. The default is df=4.
...	further arguments passed to or from other methods

Details

This function computes for every age cohort several genetic parameters. These may include

pedIBD	The average pedigree based kinship within the age cohort.
pedIBDorM	The probability that 2 alleles chosen from the age cohort are IBD or that at least one of them originates from a migrant.
pedIBDorMM	The probability that 2 alleles chosen from the age cohort are IBD or that both of them are from migrants.
pedIBDofN	The conditional probability that 2 alleles chosen from the age cohort are IBD, given that both are from native founders.
condGD	The diversity at native alleles, i.e. the conditional probability that 2 alleles chosen from the age cohort are not IBD, given that both are from native founders. Individuals born in the base-year are assumed to be unrelated.
Ne	The native effective size, quantifying how fast the diversity at native alleles is decreasing. The diversity may increase for a short time span, in which case the estimate would be NA. Use a smaller value for parameter df to get a smooth estimate.
NGE	The native genome equivalents. The parameter estimates the number of unrelated individuals that would be needed to establish a hypothetical new population that has the same condGD as the population under study. Individuals born in the base-year are assumed to be unrelated.

Value

A data frame providing for every age cohort (some of) the genetic parameters mentioned above.

Author(s)

Robin Wellmann

Examples

```
data(ExamplePed)
Pedig    <- prePed(ExamplePed, thisBreed="Hinterwaelder", lastNative=1970)
Kinships <- kinlist(pedIBD=pedIBD(Pedig), pedIBDatN=pedIBDatN(Pedig, thisBreed="Hinterwaelder"))
summary(Kinships, Pedig, tlim=c(1970,2005), histNe=150, base=1800)
```

summary.opticont *Summary Statistics and Check of Validity for Optimum Contributions.*

Description

Computation of genetic parameters and check of validity for optimum genetic contributions of the selection candidates.

Usage

```
## S3 method for class 'opticont'
summary(object, ...)
```

Arguments

object	An object from class opticont, which is usually the output of function opticont.
...	further arguments passed to or from other methods

Details

Computation of genetic parameters and check of validity for optimum genetic contributions of the selection candidates.

Value

A data frame with one row and the following columns is returned:

VarName	Name of parameter x
method	Method used for optimization
obj.fun	Value of the objective function
valid	TRUE if the side constraints are fulfilled by the optimum contributions.
KIN	The mean kinship in the offspring (one component for each kinship matrix included in the call of function opticont).
ub.KIN	The upper bound for the mean kinship in the offspring (one component for each kinship matrix included in the call of function opticont).
Div.KIN	The mean genetic diversity (=1-KIN) in the offspring (one component for each kinship matrix included in the call of function opticont).
ubM	Upper bound for the contribution of males
ubF	Upper bound for the contribution of females
ContMales	The total contribution of breeding males to the offspring (should be 0.5)
ContFemales	The total contribution of breeding females to the offspring (should be 0.5)
minCont	The minimum contribution of a breeding animal
maxContMale	The maximum contribution of a breeding male
maxContFemale	The maximum contribution of a breeding female
solver	Algorithm used for optimization
lb.VAR	The lower bound for the mean value of VAR in the offspring (one component for each numeric column VAR of data frame phen

	included in the call of function opticont).
meanVAR	The mean value of VAR in the offspring (one component for each numeric column VAR of data frame phen included in the call of function opticont).
ub.VAR	The upper bound for the mean value of VAR in the offspring (one component for each numeric column VAR of data frame phen included in the call of function opticont).

Author(s)

Robin Wellmann

Examples

```
#see ?opticont
```

---

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