Package 'ibdsim2'

July 12, 2020

Type Package

Title Simulation of Chromosomal Regions Shared by Family Members

Version 1.2

Description Simulation of segments shared identical-by-descent (IBD) by pedigree members. Using sex specific recombination rates along the human genome (Halldorsson et al. (2019) <doi:10.1126/science.aau1043>), phased chromosomes are simulated for all pedigree members. Additional features include calculation of realised IBD coefficients and plots of IBD segment distributions.

License GPL-3

URL https://github.com/magnusdv/ibdsim2

Encoding UTF-8

Language en-GB

LazyData true

Depends R (>= 3.5.0), pedtools (>= 0.9.4)

Imports Rcpp, ggplot2, glue, ribd (>= 1.1.0)

Suggests testthat

LinkingTo Rcpp

RoxygenNote 7.1.0

NeedsCompilation yes

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

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

Repository CRAN

Date/Publication 2020-07-11 22:00:10 UTC

R topics documented:

`customMap` *Custom recombination map*

Description

Create custom recombination maps for use in `ibdsim()`.

Usage

```
customMap(x)
```

Arguments

`x`	A data frame or matrix. See details for format specifications.

Details

The column names of `x` must include either

- `chrom`, `mb` and `cm` (sex-averaged map)

or

- `chrom`, `mb`, `male` and `female` (sex-specific map)

Upper-case letters are allowed in these names. The `mb` column should contain physical positions in megabases, while `cm`, `male`, `female` give the corresponding genetic position in centiMorgans.

Value

An object of class `genomeMap`.

See Also

`uniformMap()`, `loadMap()`

Examples

```
# A map including two chromosomes.
df1 = data.frame(chrom = c(1, 1, 2, 2),
                 mb = c(0, 2, 0, 5),
                 cm = c(0, 3, 0, 6))
map1 = customMap(df1)
map1

# Use columns "male" and "female" to make sex specific maps
df2 = data.frame(chrom = c(1, 1, 2, 2),
                 mb = c(0, 2, 0, 5),
                 male = c(0, 3, 0, 6),
                 female = c(0, 4, 0, 7))
map2 = customMap(df2)
map2
```

---

estimateCoeffs *Estimation of one- and two-locus relatedness coefficients*

Description

Estimate by simulation various relatedness coefficients, and two-locus versions of the same coefficients, for a given recombination rate. The current implementation covers inbreeding coefficients, kinship coefficients, IBD (kappa) coefficients between noninbred individuals, and condensed identity coefficients. These functions are primarily meant as tools for validating exact algorithms, e.g., as implemented in the `ribd` package.

Usage

```
estimateInbreeding(x, id, Nsim, Xchrom = FALSE, verbose = FALSE, ...)

estimateTwoLocusInbreeding(
  x,
  id,
  rho = NULL,
  cM = NULL,
  Nsim,
  Xchrom = FALSE,
  verbose = FALSE,
  ...
)

estimateKinship(x, ids, Nsim, Xchrom = FALSE, verbose = FALSE, ...)

estimateTwoLocusKinship(
  x,
  ids,
```

```
  rho = NULL,
  cM = NULL,
  Nsim,
  Xchrom = FALSE,
  verbose = FALSE,
  ...
)

estimateKappa(x, ids, Nsim, Xchrom = FALSE, verbose = FALSE, ...)

estimateTwoLocusKappa(
  x,
  ids,
  rho = NULL,
  cM = NULL,
  Nsim,
  Xchrom = FALSE,
  verbose = FALSE,
  ...
)

estimateIdentity(x, ids, Nsim, Xchrom = FALSE, verbose = FALSE, ...)

estimateTwoLocusIdentity(
  x,
  ids,
  rho = NULL,
  cM = NULL,
  Nsim,
  Xchrom = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

x	A pedigree in the form of a pedtools::ped() object.
id, ids	A vector of one or two ID labels.
Nsim	The number of simulations.
Xchrom	A logical indicating if the loci are X-linked (if TRUE) or autosomal (FALSE).
verbose	A logical.
...	Further arguments passed on to ibdsim(), e.g. seed.
rho	A scalar in the interval [0, 0.5]: the recombination fraction between the two loci, converted to centiMorgans using Haldane's map function: cM = -50 * log(1 - 2 * rho). Either rho or cM (but not both) must be non-NULL.
cM	A non-negative number: the genetic distance between the two loci, given in centiMorgans. Either rho or cM (but not both) must be non-NULL.

Details

In the following, let L1 and L2 denote two arbitrary autosomal loci with recombination rate ρ, and let A and B be members of the pedigree `x`.

The *two-locus inbreeding coefficient* $f_2(\rho)$ of A is defined as the probability that A is autozygous at both L1 and L2 simultaneously.

The *two-locus kinship coefficient* $\phi_2(\rho)$ of A and B is defined as the probability that a random gamete emitted from A, and a random gamete emitted from B, contain IBD alleles at both L1 and L2.

The *two-locus kappa coefficient* $\kappa_{ij}(\rho)$, for $i, j = 0, 1, 2$, of noninbred A and B, is the probability that A and B share exactly `i` alleles IBD at L1, and exactly `j` alleles IBD at L2.

The *two-locus identity coefficient* Δ_{ij}, $i, j = 1, ..., 9$ is defined for any (possibly inbred) A and B, as the probability that A and B are in identity state `i` at L1, and state `j` at L2. This uses the conventional ordering of the nine condensed identity states. For details, see for instance the GitHub page of the `ribd` package.

Value

`estimateInbreeding()`: a single probability.

`estimateTwoLocusInbreeding()`: a single probability.

`estimateKappa()`: a numeric vector of length 3, with the estimated κ coefficients.

`estimateTwoLocusKappa()`: a symmetric, numerical 3*3 matrix, with the estimated values of κ_{ij}, for $i, j = 0, 1, 2$.

`estimateIdentity()`: a numeric vector of length 9, with the estimated identity coefficients.

`estimateTwoLocusIdentity()`: a symmetric, numerical 9*9 matrix, with the estimated values of Δ_{ij}, for $i, j = 1, ..., 9$.

Examples

```
############################
### Two-locus inbreeding ###
############################

x = cousinPed(0, child = TRUE)
rho = 0.25
Nsim = 10 # Increase!
estimateTwoLocusInbreeding(x, id = 5, rho = rho, Nsim = Nsim, seed = 123)

######################################
### Two-locus kappa:               ###
### Grandparent vs half sib vs uncle ###
######################################

# These are indistinguishable with unlinked loci, see e.g.
# pages 182-183 in Egeland, Kling and Mostad (2016).
# In the following, each simulation approximation is followed
# by its exact counterpart.
```

```
rho = 0.25; R = .5 * (rho^2 + (1-rho)^2)
Nsim = 10 # Should be increased to at least 10000

# Grandparent/grandchild
G = linearPed(2); G.ids = c(1,5); # plot(G, shaded = G.ids)
estimateTwoLocusKappa(G, G.ids, rho = rho, Nsim = Nsim, seed = 123)[2,2]
.5*(1-rho) # exact

# Half sibs
H = halfSibPed(); H.ids = c(4,5); # plot(H, shaded = H.ids)
estimateTwoLocusKappa(H, H.ids, rho = rho, Nsim = Nsim, seed = 123)[2,2]
R # exact

# Uncle
U = cousinPed(0, removal = 1); U.ids = c(3,6); # plot(U, shaded = U.ids)
estimateTwoLocusKappa(U, U.ids, rho = rho, Nsim = Nsim, seed = 123)[2,2]
(1-rho) * R + rho/4 # exact

# Exact calculations by ribd:
# ribd::twoLocusIBD(G, G.ids, rho = rho, coefs = "k11")
# ribd::twoLocusIBD(H, H.ids, rho = rho, coefs = "k11")
# ribd::twoLocusIBD(U, U.ids, rho = rho, coefs = "k11")

##########################
### Two-locus Jacquard ###
##########################

x = fullSibMating(1)
rho = 0.25
Nsim = 10 # (increase to at least 10000)

estimateTwoLocusIdentity(x, ids = 5:6, rho = rho, Nsim = Nsim, seed = 123)

# Exact by ribd:
# ribd::twoLocusIdentity(x, ids = 5:6, rho = rho)
```

`haploDraw` *Draw haplotypes onto a pedigree plot*

Description

Visualise the IBD pattern of a single chromosome, by drawing haplotypes onto the pedigree.

Usage

```
haploDraw(
  x,
  ibd,
```

```
  chrom = NULL,
  pos = 1,
  cols = NULL,
  height = 4,
  width = 0.5,
  sep = 0.75,
  dist = 1.5,
  ...
)
```

Arguments

x	A ped object.
ibd	A genomeSim object.
chrom	A chromosome number, needed if ibd contains data from multiple chromosomes.
pos	A vector recycled to the length of labels(x), indicating where haplotypes should be drawn relative to the pedigree symbols: 0 = no haplotypes; 1 = below; 2 = left; 3 = above; 4 = right. By default, all are placed below.
cols	A colour vector corresponding to the alleles in ibd.
height	The haplotype height divided by the height of a pedigree symbol.
width	The haplotype width divided by the width of a pedigree symbol.
sep	The separation between haplotypes within a pair, given as a fraction of width.
dist	The distance between pedigree symbols and the closest haplotype, given as a fraction of width.
...	Arguments passed on to plot.ped().

Value

None.

Examples

```
op = par()

##############################
# Example 1: A family quartet #
##############################

x = nuclearPed(2)
s = ibdsim(x, N = 1, map = uniformMap(M = 1), seed = 4276)
s[[1]]

haploDraw(x, s[[1]], pos = c(2,4,1,1), cols = c(3,7,2,4),
          margin = c(3, 5, 3, 5), cex = 1.2)


##########################
```

```
# Example 2: Autozygosity #
###########################

x = halfCousinPed(0, child = TRUE)
s = ibdsim(x, N = 1, map = uniformMap(M = 1),
           skipRecomb = spouses(x, 1), seed = 19499)
s[[1]]

# Gray colour (8) for irrelevant founder alleles
haploDraw(x, s[[1]], pos = c(1,0,2,0,4,4),
          cols = c(3,7,8,8,8,8), margin = c(2, 2, 2, 2))


# Restore graphics parameters
par(op)
```

---

ibdsim *IBD simulation*

Description

This is the main function of the package, simulating the recombination process in each meioses of a pedigree. The output summarises the IBD segments between all or a subset of individuals.

Usage

```
ibdsim(
  x,
  N = 1,
  ids = labels(x),
  map = "decode",
  model = c("chi", "haldane"),
  skipRecomb = NULL,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

x	A pedtools::ped() object.
N	A positive integer indicating the number of simulations.
ids	A subset of pedigree members whose IBD sharing should be analysed. If NULL, all members are included.
map	The genetic map to be used in the simulations: Allowed values are: • a genomeMap object, typically produced by loadMap() • a single chromMap object, for instance as produced by uniformMap()

- a character, which is passed on to `loadMap()` with default parameters. Currently the only valid option is "decode19" (or abbreviations of this).

Default: "decode19".

`model`	Either "chi" or "haldane", indicating the statistical model for recombination (see details). Default: "chi".
`skipRecomb`	A vector of ID labels indicating individuals whose meioses should be simulated without recombination. (Each child will then receive a random strand of each chromosome.) The default action is to skip recombination in founders who are uninformative for IBD sharing in the `ids` individuals.
`seed`	An integer to be passed on to `set.seed()`).
`verbose`	A logical.

Details

Each simulation starts by unique alleles (labelled 1, 2, ...) being distributed to the pedigree founders. In each meiosis, homologue chromosomes are made to recombine according to the value of `model`:

- `model = "haldane"`: In this model, crossover events are modelled as a Poisson process along each chromosome.
- `model = "chi"` (default): This uses a renewal process along the four-strand bundle, with waiting times following a chi square distribution.

Recombination rates along each chromosome are determined by the `map` parameter. The default value ("decode19") loads a thinned version of the recombination map of the human genome published by Halldorsson et al (2019).

In many applications, the fine-scale default map is not necessary, and should be replaced by simpler maps with constant recombination rates. See `uniformMap()` and `loadMap()` for ways to produce such maps.

Value

A list of `N` objects of class `genomeSim`.

A `genomeSim` object is essentially a numerical matrix describing the allele flow through the pedigree in a single simulated. Each row corresponds to a chromosomal segment. The first 4 columns describe the segment (chromosome, start, end, length), and are followed by two columns (paternal allele, maternal allele) for each of the `ids` individuals.

If `ids` has length 1, a column named "Aut" is added, whose entries are 1 for autozygous segments and 0 otherwise.

If `ids` has length 2, two columns are added:

- `IBD` : The IBD status of each segment (= number of alleles shared identical by descent). For a given segment, the IBD status is either 0, 1, 2 or NA. If either individual is inbred, they may be autozygous in a segment, in which case the IBD status is reported as NA. With inbred individuals the `Sigma` column (see below) is more informative than the `IBD` column.
- `Sigma` : The condensed identity ("Jacquard") state of each segment, given as an integer in the range 1-9. The numbers correspond to the standard ordering of the condensed states. In particular, for non-inbred individuals the states 9, 8, 7 correspond to IBD status 0, 1, 2 respectively.

References

Halldorsson et al. *Characterizing mutagenic effects of recombination through a sequence-level genetic map.* Science 363, no. 6425 (2019).

Examples

```
hs = halfSibPed()
ibdsim(hs, N = 2, map = uniformMap(M = 1), ids = 4:5)

# Full sib mating: all 9 states are possible
x = fullSibMating(1)
sim = ibdsim(x, N = 1, ids = 5:6, map = uniformMap(M = 10), seed = 1)
s = sim[[1]]
stopifnot(setequal(s[, 'Sigma'], 1:9))
```

`ibdsim2` *ibdsim2: Simulation of chromosomal regions shared by family members*

Description

Simulation of segments shared identical-by-descent (IBD) by pedigree members. Using sex specific recombination rates along the human genome (Halldorsson et al., 2019), phased chromosomes are simulated for all pedigree members. Additional features include calculation of realised IBD coefficients and IBD segment distribution plots.

References

Halldorsson et al. *Characterizing mutagenic effects of recombination through a sequence-level genetic map.* Science 363, no. 6425 (2019) doi: 10.1126/science.aau1043

`loadMap` *Load a built-in genetic map*

Description

This function loads one of the built-in genetic maps. Currently, the available map is based on the publication by Halldorsson et al. (2019).

Usage

```
loadMap(map = "decode19", chrom = 1:22, uniform = FALSE, sexAverage = FALSE)
```

Arguments

map	The name of the wanted map, possibly abbreviated. Currently, the only valid choice is "decode19" (default).
chrom	A numeric vector indicating which chromosomes to load. Default: 1:22 (the autosomes).
uniform	A logical. If FALSE (default), the complete inhomogeneous map is used. If TRUE, a uniform version of the same map is produced, i.e. with the correct lengths, but constant recombination rate along each chromosome.
sexAverage	A logical, by default FALSE. If TRUE, a sex-averaged map is returned, with equal recombination rates for males and females.

Details

For reasons of speed and efficiency, the built-in map is a thinned version of the published map (Halldorsson et al., 2019), keeping around 60 000 data points.

By setting uniform = TRUE, a uniform version of the map is returned, in which each chromosome has the same genetic lengths as in the original, but with constant recombination rates. This gives much faster simulations and may be preferable in some applications.

Value

An object of class genomeMap.

References

Halldorsson et al. *Characterizing mutagenic effects of recombination through a sequence-level genetic map.* Science 363, no. 6425 (2019).

See Also

uniformMap(), customMap()

Examples

```
# By default, the complete map of all 22 autosomes is returned
loadMap()

# Uniform version
m = loadMap(uniform = TRUE)

# Check chromosome 1
m1 = m[[1]]
m1$male
m1$female
```

`plotSegmentDistribution` *Scatter plots of IBD segment distributions*

Description

Visualise and compare count/length distributions of IBD segments. Two types are currently implemented: Segments of autozygosity (for a single person) and segments with (pairwise) IBD state 1.

Usage

```
plotSegmentDistribution(
  ...,
  type = c("autozygosity", "ibd1"),
  ids = NULL,
  labels = NULL,
  col = NULL,
  shape = 1,
  alpha = 1,
  ellipses = TRUE,
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  legendInside = TRUE
)
```

Arguments

`...`	One or several objects of class `genomeSimList`, typically created by `ibdsim()`. They can be entered separately or as a `list`.
`type`	A string indicating which segments should be plotted. Currently, the allowed entries are "autozygosity" and "ibd1".
`ids`	A list of the same length as `...`, where each entry contains one or two ID labels (depending on `type`). By default (NULL), these labels are extracted from the inputs in `...`. Two other short-cuts are possible: If a single vector is given, it is repeated for all pedigrees. Finally, if `ids` is the word "leaves" then `pedtools::leaves()` is used to extract labels in each pedigree.
`labels`	An optional character vector of labels used in the legend. If NULL, the labels are taken from `names(...)`.
`col`	An optional colour vector of the same length as `...`.
`shape`	A vector with point shapes, of the same length as `...`.
`alpha`	A transparency parameter for the scatter points.
`ellipses`	A logical: Should confidence ellipses be added to the plot?

`title, xlab, ylab`
: Title and axis labels.

`legendInside`
: A logical controlling the legend placement.

Details

This function takes as input one or several complete outputs from the `ibdsim()`, and produces a scatter plot of the number and average length of IBD segments from each.

Contour curves are added to plot, corresponding to the theoretical/pedigree-based values: either inbreeding coefficients (if `type = "autozygosity"`) or κ_1 (if `type = "ibd1"`).

Examples

```
# Simulation parameters used in the below examples.
map = uniformMap(M = 10)   # recombination map
N = 5                      # number of sims

# For more realistic results, replace with e.g.:
# map = loadMap("decode19")
# N = 1000


######################################################################
# EXAMPLE 1
# Comparison of IBD segment distributions
# between paternal and maternal half siblings.
######################################################################

# Define the pedigrees
xPat = halfSibPed()
xMat = swapSex(xPat, 1)

simPat = ibdsim(xPat, N = N, map = map)
simMat = ibdsim(xMat, N = N, map = map)

# By default, the IBD segments of the "leaves" are computed and plotted
plotSegmentDistribution(simPat, simMat, type = "ibd1", ids = 4:5,
                        labels = c("HSpat", "HSmat"))

######################################################################
# EXAMPLE 2
# Half siblings vs half uncle vs grandparent/grandchild
######################################################################

# Only one pedigree needed here
x = addSon(halfSibPed(), 5)

s = ibdsim(x, N = N, map = map)

# Indicate the pairs explicitly this time.
ids = list(HS = 4:5, HU = c(4,7), GR = c(1,7))
```

```
# List names are used as labels in the plot
plotSegmentDistribution(s, type = "ibd1", ids = ids, shape = 1:3)


#################################################################
# EXAMPLE 3
# Comparison of autozygosity distributions in various individuals
# with the same expected inbreeding coefficient (f = 1/8)
#################################################################

G = swapSex(linearPed(2), 5)           # grandfather/granddaughter
G = addChildren(G, 1, 5, 1)
HSpat = swapSex(halfSibPed(), 5)       # paternal half sibs
HSpat = addChildren(HSpat, 4, 5, 1)
HSmat = swapSex(HSpat, 1)              # maternal half sibs
QHFC = quadHalfFirstCousins()          # quad half first cousins
QHFC = addChildren(QHFC, 9, 10, nch = 1)

peds = list(G = G, HSpat = HSpat, HSmat = HSmat, QHFC = QHFC)
plotPedList(peds, newdev = TRUE)
dev.off()

# Simulations
s = lapply(peds, function(p)
  ibdsim(p, N = N, ids = leaves(p), verbose = FALSE, map = map))

# Plot distributions
plotSegmentDistribution(s, type = "autoz", title = "Autozygous segments")
```

`profileSimIBD` *Simulate markers on a given IBD pattern*

Description

This function simulates genotypes for a set of markers, conditional on a specific underlying IBD pattern.

Usage

```
profileSimIBD(x, ibdpattern, ids = NULL, markers = NULL, seed = NULL)
```

Arguments

`x`	A ped object.
`ibdpattern`	A `genomeSim()` object, typically created by `ibdsim()`. (See Examples).
`ids`	A vector of ID labels. If NULL, all members of `x` are included.
`markers`	A vector with names of indices of markers attached to `x`.
`seed`	An integer seed for the random number generator.

Details

It should be noted that the only *random* part of this function is the selection of founder alleles for each marker. Given those, all other genotypes in the pedigree are determined by the underlying IBD pattern.

Value

An object similar to `x`. but with simulated genotypes.

See Also

`ibdsim()`

Examples

```
# A pedigree with two siblings
x = nuclearPed(2)

# Attach 3 linked markers on chromosome 1
pos = c(20, 50, 70)   # marker positions in megabases
mlist = lapply(pos, function(i)
  marker(x, alleles = letters[1:10], chrom = 1, posMb = i))
x = setMarkers(x, mlist)

# Simulate the underlying IBD pattern in the pedigree
s = ibdsim(x, 1, map = uniformMap(M = 1, chrom = 1), seed = 123)[[1]]

# Simulate genotypes for the sibs conditional on the given IBD pattern
profileSimIBD(x, s, ids = 3:4, seed = 123)

# With a different seed
profileSimIBD(x, s, ids = 3:4, seed = 124)
```

`realised` *Realised relatedness*

Description

Compute the realised values of various pedigree coefficients, from simulated data. The current implementation covers inbreeding coefficients for single pedigree members, and kinship, kappa and condensed identity coefficients for pairwise relationships.

Usage

```
realisedInbreeding(sims, id = NULL)

realisedKinship(sims, ids = NULL)
```

```
realisedKappa(sims, ids = NULL)

realisedIdentity(sims, ids = NULL)
```

Arguments

sims	A list of genome simulations, as output by ibdsim().
id, ids	A vector with one or two ID labels.

Details

The inbreeding coefficient f of a pedigree member is defined as the probability of autozygosity (homozygous for alleles that are identical by descent) in a random autosomal locus. Equivalently, the inbreeding coefficient is the *expected* autozygous proportion of the autosomal chromosomes.

The *realised* inbreeding coefficient f_R in a given individual is the actual fraction of the autosomes covered by autozygous segments. Because of the stochastic nature of meiotic recombination, this may deviate substantially from the pedigree-based expectation.

Similarly, the pedigree-based IBD coefficients $\kappa_0, \kappa_1, \kappa_2$ of noninbred pairs of individuals have realised counterparts. For any given pair of individuals we define k_i to be the actual fraction of the autosome where the individuals share exactly i alleles IBD, where $i = 0, 1, 2$.

Finally, we can do the same thing for each of the nine condensed identity coefficients of Jacquard. For each $i = 1, ..., 9$ we define D_i the be the fraction of the autosome where a given pair of individuals are in identity state i. This uses the conventional ordering of the nine condensed identity states; see for instance the ribd GitHub page.

Examples

```
# Realised IBD coefficients between full siblings
x = nuclearPed(2)
s = ibdsim(x, N = 2) # increase N
realisedKappa(s, ids = 3:4)

###########

# Realised inbreeding coefficients, child of first cousins
x = cousinPed(1, child = TRUE)
s = ibdsim(x, N = 2) # increase N
realisedInbreeding(s, id = 9)

# Same data: realised kinship coefficients between the parents
realisedKinship(s, ids = parents(x, 9))

###########

# Realised identity coefficients after full sib mating
x = fullSibMating(1)
s = ibdsim(x, N = 2) # increase N
realisedIdentity(s, ids = 5:6)
```

uniformMap *Uniform recombination maps*

Description

Create a uniform recombination map of a given length.

Usage

```
uniformMap(Mb = NULL, cM = NULL, M = NULL, cmPerMb = 1, chrom = 1)
```

Arguments

Mb	Map length in megabases.
cM	Map length in centiMorgan.
M	Map length in Morgan.
cmPerMb	A positive number; the cM/Mb ratio.
chrom	A chromosome label.

Value

An object of class chromMap, which is a list of two matrices, named "male" and "female".

See Also

loadMap(), customMap()

Examples

```
uniformMap(M = 1)

m = uniformMap(Mb = 1, cM = 2:3)
```

zeroIBD *Probability of zero IBD*

Description

Estimate the probability of no IBD sharing in a pairwise relationship.

Usage

```
zeroIBD(sims, ids = NULL, threshold = 0)
```

Arguments

sims	A list of genome simulations, as output by ibdsim().
ids	A vector with two ID labels. If NULL (default), these are deduced from the sims object.
threshold	A nonnegative number (default:0). Only IBD segments longer than this are included in the computation.

Value

A list with the following two entries:

- zeroprob: The fraction of sims in which ids have no IBD sharing
- stErr: The standard error of zeroprob

Examples

```
###
# The following example computes the probability of
# no IBD sharing between a pair of fourth cousins.
# We also show how the probability is affected by
# truncation, i.e., ignoring short segments.
###

# Define the pedigree
x = cousinPed(4)
cous = leaves(x)

# Simulate (increase N!)
s = ibdsim(x, N = 10)

# Probability of zero ibd segments. (By default all segs are used)
zeroIBD(s, ids = cous)

# Re-compute with positive threshold
zeroIBD(s, ids = cous, threshold = 1)
```

Index