Package: ibdsegments (via r-universe)

June 3, 2026

Title Identity by Descent Probability in Pedigrees

Version 1.0.1

Description Identity by Descent (IBD) distributions in pedigrees. A Hidden Markov Model is used to compute identity coefficients, simulate IBD segments and to derive the distribution of total IBD sharing and segment count across chromosomes. The methods are applied in Kruijver (2025) <doi:10.3390/genes16050492>. The probability that the total IBD sharing is zero can be computed using the method of Donnelly (1983) <doi:10.1016/0040-5809(83)90004-7>.

License GPL (>= 2)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

LinkingTo Rcpp

Imports Rcpp, pedtools, expm

Suggests testthat (>= 3.0.0), ribd, ggplot2

Config/testthat/edition 3

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

Date/Publication 2025-04-28 09:38:35 UTC

RemoteUrl https://github.com/mkruijver/ibdsegments

RemoteRef HEAD

RemoteSha 04e5b58dbe5aa8b1fe39b6f5aedf09776ae6d765

Contents

`convolve_total_ibd_dists` *Convolve IBD distributions to obtain the distribution of the sum*

Description

Chromosome-wise IBD distributions obtained from the `total_ibd_dist` function can be convoluted manually using the `convolve_total_ibd_dists` function. This allows finer control of the procedure by controlling the number of gridpoints used in the FFT and the threshold for point masses to be retained.

Usage

```
convolve_total_ibd_dists(
  ...,
  point_mass_eps = 1e-09,
  number_of_gridpoints_exponent = 12
)
```

Arguments

`...`	ibd dists.
`point_mass_eps`	Point masses smaller than this value are discarded.
`number_of_gridpoints_exponent`	Default is 12.

Details

The `convolve_total_ibd_dists` implements a convolution procedure based on a self contained variant of the recipe implemented in the `distr` package adapted to this use case. In particular, the IBD distribution for one chromosome is often a mixture of two point masses and continuous with finite support otherwise. Convolution increases the number of point masses and decreases their probability mass. This function assumes that small point masses are not of specific interest and will discard the point masses when the probability mass is smaller than `point_mass_eps` with a default

value of `1e-9`. For typical pedigree relationships this means that the IBD distribution of more than a few chromosomes is treated as a continuous distribution admitting a density function.

The `number_of_gridpoints_exponent` controls the density of gridpoints in the FFT. By default this is `12` which means that `2^12=4096` gridpoints are used. Increasing this parameter improves accuracy at the cost of increased runtime and memory use.

The point masses are of particular interest in some applications. For example, the probability that no part of the autosomal genome is inherited may be of interest. In that case, the point masses should not be discarded and the `d_cibd` function may be used. See the example below, for further details.

Value

Object of class `total_ibd_dist`

Examples

```
## Convolution of IBD distributions for half siblings at chromosome 1 and 2

# define half sib pedigree
ped_hs <- pedtools::halfSibPed()

# obtain chromosome-wise distributions
d1 <- total_ibd_dist(pedigree = ped_hs, chromosome_length = 267.77)
d2 <- total_ibd_dist(pedigree = ped_hs, chromosome_length = 251.73)

convolve_total_ibd_dists(d1, d2) # 4 point masses

## Accuracy of convolution depends on number of gridpoints

L <- c(267.77, 251.73, 218.31, 202.89, 197.08, 186.02, 178.4, 161.54,
       157.35, 169.28, 154.5, 165.49, 127.23, 116, 117.32, 126.59, 129.53,
       116.52, 106.35, 107.76, 62.88, 70.84)

# obtain chromosome-wise distributions for half siblings
hs <- total_ibd_dist(pedigree = ped_hs,
                     chromosome_length = L, convolve = FALSE)

# obtain moments of the sum by summing the moments..
mean_hat <- sum(sapply(hs, E))
sd_hat <- sqrt(sum(sapply(hs, var)))

# .. or by summing the distributions with varying numbers of gridpoints
k <- 6:10
sd_hat_by_k <- sapply(k, function(k) sd(convolve_total_ibd_dists(hs,
                                        number_of_gridpoints_exponent = k)))
mean_hat_by_k <- sapply(k, function(k) E(convolve_total_ibd_dists(hs,
                                        number_of_gridpoints_exponent = k)))

plot(k, mean_hat_by_k)
abline(h = mean_hat, lty = 2)
```

```
plot(k, sd_hat_by_k)
abline(h = sd_hat, lty = 2)
```

---

d *Probability function for IBD Distribution Objects*

Description

Returns the probability function of a total IBD or segment count distribution.

Usage

```
d(x, ...)
```

Arguments

x	An object of class `total_ibd_dist` or `segment_count_dist`.
...	Additional arguments (not used).

Value

A numeric value.

---

d_cibd *Probability density of continuously observed IBD segment lengths for pedigree members*

Description

The `d_cibd` function computes the probability density of observed IBD segments on a chromosome.

Usage

```
d_cibd(
  x,
  ibd,
  pedigree,
  ids = pedtools::leaves(pedigree),
  states = "ibd",
  log10 = FALSE
)
```

Arguments

x	Numeric vector with lengths of segments (centiMorgan) or result from r_cibd.
ibd	Integer vector with IBD states in segments if x is not a result from r_cibd. Taking values 0, 1, 2 for states = "ibd" or states = "kappa", 1, ..., 9 for states="identity" and 1, ..., 15 for states = "detailed".
pedigree	Pedigree in pedtools::ped form.
ids	Ids for which IBD is observed. Default is pedtools::leaves(pedigree).
states	One of "ibd" (default), "kappa", "identity" or "detailed".
log10	Should the log10 probability density be returned? Default is FALSE.

Value

Numeric

Examples

```
ped_fs <- pedtools::nuclearPed(nch = 2)

# probability that full siblings are double IBD (kappa2)
d_cibd(x = 0, ibd = 2, ped_fs)

# full siblings are double IBD and remain so for 100 cM
d_cibd(x = 100, ibd = 2, ped_fs)

# full siblings are double IBD for 50 cM,
# then single IBD for 50 cM
d_cibd(x = c(50, 50), ibd = c(2, 1), ped_fs)

# full siblings are double IBD, remain so for 100 cM
# and no longer
d_cibd(x = c(100, 0), ibd = c(2, 1), ped_fs)

## probability density of IBD segment length for first cousins on an infinite chromosome
ped_fc <- pedtools::cousinPed()
# first compute the probability of IBD
k1_fc <- d_ibd(ibd = 1, ped_fc)
# density of segment length
f <- Vectorize(function(l) d_cibd(x = c(l,0), ibd = c(1, 0), ped_fc) / k1_fc)

curve(f, 0, 300)

# f is a probability density (integrates to 1)
integrate(f, 0, Inf)

# for full siblings, how does the chance of remaining double IBD
# depend on the segment length?
cM <- seq(from = 0, to = 100, length = 200)
pr_2ibd <- sapply(cM, d_cibd, 2, ped_fs) / d_ibd(2, ped_fs)

plot(cM, pr_2ibd, type="l")
```

```
# since there are four meioses, the sojourn time in this IBD state
# follows an exponential distribution with rate 0.04
stopifnot(all.equal(pr_2ibd, pexp(cM, rate = 0.04, lower.tail = FALSE)))

## Using the output from r_cibd directly to compute evidential value
## of IBD segments on chromosome for distinguishing first and second cousins
ped_c1 <- pedtools::cousinPed()
ped_c2 <- pedtools::cousinPed(degree = 2)

r_c1 <- r_cibd(n = 1e2, pedigree = ped_c1)

lr <- d_cibd(r_c1, pedigree = ped_c1) / d_cibd(r_c1, pedigree = ped_c2)

hist(log10(lr))
```

---

d_ibd *Compute probability of IBD for pedigree members*

Description

The d_ibd function computes the likelihood of IBD for one position or multiple linked markers on the same chromosome.

Usage

```
d_ibd(
  ibd,
  pedigree,
  ids = pedtools::leaves(pedigree),
  recombination_rate_by_locus = numeric(),
  states = "ibd",
  log10 = FALSE
)
```

Arguments

ibd	Integer vector. Taking values 0, 1, 2 for states = "ibd" or states = "kappa", 1, ..., 9 for states="identity" and 1, ..., 15 for states = "detailed".
pedigree	Pedigree in pedtools::ped form.
ids	Ids for which IBD is observed. Defaults to pedtools::leaves(pedigree).
recombination_rate_by_locus	Numeric vector with length one shorter than ibd.
states	One of "ibd" (default), "kappa", "identity" or "detailed".
log10	Should the log10 likelihood be returned? Default is FALSE.

Value

Numeric

Examples

```
# Compute kappa0, kappa1, kappa2 for full siblings
ped_fs <- pedtools::nuclearPed(nch = 2)

k0 <- d_ibd(ibd = 0, pedigree = ped_fs)
k1 <- d_ibd(ibd = 1, pedigree = ped_fs)
k2 <- d_ibd(ibd = 2, pedigree = ped_fs)
c(k0, k1, k2)

stopifnot(identical(c(k0, k1, k2), c(0.25, 0.5, 0.25)))

# Compute kappa00 for two tightly linked loci
d_ibd(c(0,0), pedigree = ped_fs,
       recombination_rate_by_locus = c(0.01))

# or 100 tightly linked loci
d_ibd(rep(0, 100), pedigree = ped_fs,
               recombination_rate_by_locus = c(rep(0.01, 99)))

# Jacquard's 9 condensed and 15 detailed identity coefficients
ped_fs_mating <- pedtools::fullSibMating(1)

sapply(1:9, d_ibd, pedigree = ped_fs_mating, states = "identity")
sapply(1:15, d_ibd, pedigree = ped_fs_mating, states = "detailed")
```

E *Expectation for IBD Distribution Objects*

Description

Computes the expectation (mean) of a total IBD or segment count distribution.

Usage

```
E(x, ...)
```

Arguments

x	An object of class `total_ibd_dist` or `segment_count_dist`.
...	Additional arguments (not used).

Value

A numeric value.

inheritance_pattern *Inheritance pattern for inheritance vectors*

Description

The inheritance_pattern function determines the inheritance pattern for all pedigree members by dropping the founder allele labels down the pedigree according to the IBD vector v.

Usage

```
inheritance_pattern(inheritance_space, v)
```

Arguments

inheritance_space	Output of inheritance_space.
v	Integer.

Value

Data frame describing inheritance pattern and IBD state for each v.

Examples

```
ped_fs <- pedtools::nuclearPed(nch = 2)
i <- inheritance_space(ped_fs, ids = 3:4)

# show the inheritance pattern and IBD state for all canonical IBD vectors
inheritance_pattern(i, v = 0:3)

# without exploiting founder symmetry
i2 <- inheritance_space(ped_fs, ids = 3:4, exploit_symmetries = FALSE)
inheritance_pattern(i2, v = 0:15)
```

inheritance_space *Inheritance space for pedigree*

Description

The inheritance_space function determines the space of IBD vectors for a pedigree. This is mostly for internal use but may be interesting by itself.

Usage

```
inheritance_space(pedigree, ids, states = "ibd", exploit_symmetries = TRUE)
```

Arguments

pedigree	Pedigree in pedtools::ped form.
ids	Ids for which IBD is observed. Default is pedtools::leaves(pedigree).
states	One of "ibd" (default), "kappa", "identity" or "detailed".
exploit_symmetries	Should symmetries be used to reduce to state space? This can be set to FALSE for debugging purposes.

Value

Object of class inheritance_space.

Examples

```
# set up inheritance space for half sib pedigree
i <- inheritance_space(pedigree = pedtools::halfSibPed())

# since there are 2 non-founders, there are 2^4 IBD vectors
# but only 2 distinct states are considered because of symmetries
i

# pry into the internals to see individual pedigree transmissions
i$transmissions
```

---

multi_ibd_patterns *Compute probabilities of minimal multi-id IBD patterns*

Description

For two full siblings at one locus, it is well known that there are three distinct minimal IBD patterns with probabilities 0.25, 0.5 and 0.25. The ribd::multiPersonIBD function generalises the computation of these patterns and their probabilities to more than two ids The multi_ibd_patterns function further generalises the computation to patterns across multiple loci.

Usage

```
multi_ibd_patterns(
  pedigree,
  ids = pedtools::leaves(pedigree),
  recombination_rate_by_locus = numeric()
)
```

Arguments

pedigree	Pedigree in pedtools::ped form.
ids	Ids for which IBD is observed. Defaults to pedtools::leaves(pedigree).
recombination_rate_by_locus	Optionally a numeric vector with recombination rates.

Value

DataFrame

Examples

```
# Compute IBD patterns for two full siblings...
multi_ibd_patterns(pedtools::nuclearPed(nch = 2))

# ... and the generalisation to three siblings
multi_ibd_patterns(pedtools::nuclearPed(nch = 3))

# Two full siblings at two tightly linked loci
multi_ibd_patterns(pedtools::nuclearPed(nch = 2),
                   recombination_rate_by_locus = 0.01)
```

---

pr_0_total_ibd *Probability of no IBD across one or more chromosomes*

Description

Donnelly (1983) studies the probability that relatives of certain types have no segments of a chromosome in common and provides expressions that can be efficiently computed.

Usage

```
pr_0_total_ibd(
  relationship_type = c("cousin", "grandparent", "halfsib", "uncle"),
  degree,
  removal,
  removal1,
  removal2,
  chromosome_length,
  log10 = FALSE
)
```

Arguments

relationship_type	One of "cousin", "halfsib", "grandparent" or "uncle".
degree	See details.
removal	See details.
removal1	See details.
removal2	See details.
chromosome_length	Default is 267.77 cM (an estimate of the length of chromosome 1).
log10	Should the log10 probability be returned? Default is FALSE.

Details

Types of relationships supported are:

`cousin` Use `degree = a` and `removal = b` for a'th cousins b times removed with degree at least one. Default is `degree = 1`.

`halfsib` Use `removal = 0` (default) for half-siblings and `removal = a` for half-cousins a times removed.

`grandparent` `degree = 1` (default) for grandparents, 2 for great-grandparents and so on.

`uncle` Use `degree = 0` (default) for uncle and `degree = d` for great^d uncle

Value

Numeric

References

Donnelly K. P. (1983). The probability that related individuals share some section of genome identical by descent. Theoretical population biology, 23(1), 34–63. https://doi.org/10.1016/0040-5809(83)90004-7

See Also

pr_all_genes_survive for the probability that all autosomal genes are passed on to the next generation (offspring column in Table 1 of Donnelly (1983))

Examples

```
## Cousin-type: third cousins on a chromosome of length 100 cM
degree <- 3

p0_3C <- pr_0_total_ibd("cousin", degree = 3, chromosome_length = 100)
p0_3C

# verify
p0_3C_manual <- d_cibd(x = 100, ibd = 0,
                       pedigree = pedtools::cousinPed(degree = 3))

p0_3C_manual
stopifnot(all.equal(p0_3C, p0_3C_manual))

## Half-sib type: half-cousins twice removed
p0_H1C_2R <- pr_0_total_ibd("halfsib",
                            degree = 1, removal = 2, chromosome_length = 100)
p0_H1C_2R

p0_H1C_2R_manual <- d_cibd(x = 100, ibd = 0,
                           pedigree = pedtools::halfCousinPed(removal = 2))

p0_H1C_2R_manual
```

```
stopifnot(all.equal(p0_H1C_2R, p0_H1C_2R_manual))

## Grandparent-type: great grandparents (degree = 2)
p0_GGP <- pr_0_total_ibd("grandparent", degree = 2, chromosome_length = 100)
p0_GGP

# GGP is a third generation ancestor so n = 3
p0_GGP_manual <- d_cibd(x = 100, ibd = 0,
                    pedigree = pedtools::linearPed(n = 3),
                    ids = c(1, pedtools::leaves(pedtools::linearPed(n = 3))))

p0_GGP_manual

stopifnot(all.equal(p0_GGP, p0_GGP_manual))

## Uncle-type: degree = 0 for uncle
p0_AV <- pr_0_total_ibd("uncle", chromosome_length = 100)
p0_AV

p0_AV_manual <- d_cibd(x = 100, ibd = 0, pedigree = pedtools::avuncularPed())

p0_AV_manual

stopifnot(all.equal(p0_AV, p0_AV_manual))

## Reproduce Table 1 of Donnelly (1983)
# (historic) chromosome lengths (cM) used in Donnelly (1983)
L <- 33 * c(9.12, 8.53, 7.16, 6.59, 6.15, 5.87, 5.31, 4.92, 4.81, 4.71, 4.60,
            4.47, 3.56, 3.60, 3.40, 3.20, 3.12, 2.72, 2.48, 2.27, 1.77, 1.64)
k <- 4:15

tab1 <- data.frame(k=k)

tab1$cousin <- pr_0_total_ibd(relationship_type = "cousin",
                              degree = rep(2:7, each = 2),
                              removal = rep(0:1, times = 6),
                              chromosome_length = L)

tab1$uncle <- pr_0_total_ibd(relationship_type = "uncle",
                             degree = k - 1, chromosome_length = L)

# Note the removal on one side only
tab1$halfsib <- pr_0_total_ibd(relationship_type = "halfsib",
                               removal1 = k - 1,
                               removal2 = rep(0, length(k)),
                               chromosome_length = L)

# Poisson approximation
tab1$`exp(-33 * k / 2^k)` <- exp(-33 * k / 2^k)

# Note that k corresponds to great^k grandparent,
# i.e. a (k+2)'th generation ancestor
# (not great^(k-1) and (k+1)'th generation ancestor as printed)
```

```
tab1$grandparent <- pr_0_total_ibd(relationship_type = "grandparent",
                                   degree = k, chromosome_length = L)

tab1
```

`pr_all_genes_survive` *Probability of passing down all autosomal genes to offspring*

Description

Donnelly (1983) presents a way to efficiently calculate the probability of passing down all genes to the next generation. This includes both the paternally and maternally inherited genes, so at least two offspring are needed for this probability to be non-zero.

Usage

```
pr_all_genes_survive(
  number_of_children,
  chromosome_length = 267.66,
  log10 = FALSE
)
```

Arguments

`number_of_children`	How many offspring? Result is vectorised over this parameter.
`chromosome_length`	Default is 267.77 cM (an estimate of the length of chromosome 1). `length(chromosome_length)>1`, the probability of passing down all chromosomes will be computed.
`log10`	Should the log10 probability be returned? Default is `FALSE`.

Value

Numeric

References

Donnelly K. P. (1983). The probability that related individuals share some section of genome identical by descent. Theoretical population biology, 23(1), 34–63. https://doi.org/10.1016/0040-5809(83)90004-7

Examples

```
## Example from Donnelly (1983)
# (historic) chromosome lengths (cM) used in Donnelly (1983)
L <- 33 * c(9.12, 8.53, 7.16, 6.59, 6.15, 5.87, 5.31, 4.92, 4.81, 4.71, 4.60,
            4.47, 3.56, 3.60, 3.40, 3.20, 3.12, 2.72, 2.48, 2.27, 1.77, 1.64)

# Reproduce the "Offspring" column in Table 1
number_of_children <- 1:16
pr_survival <- pr_all_genes_survive(number_of_children, chromosome_length = L)

# Plot the results (compare to Fig. 10)
plot(number_of_children, pr_survival)

# Add the Poisson approximation
k <- 2:17 - 1
lines(k+1, exp(-k*33 / (2^k)), lty=2)

## Example using general purpose methods
# The probability of passing down all genes to k offspring is equal to the
# probability that the joint IBD state of k half siblings is 0 everywhere -
# there is no point where they all inherited the same DNA from the common
# parent.

# Define a function to create a pedigree of half siblings
ped_halfsibs <- function(number_of_half_sibs){
  ped <- pedtools::nuclearPed(nch = 1)
  for (k in 2:number_of_half_sibs) {
    ped <- pedtools::addChild(ped, parents = c(1), verbose = FALSE)
  }
  ped
}

# Compute the probability that a chromosome of length 100 cm survives
# if the next generation consists of 5 children
p1 <- pr_all_genes_survive(number_of_children = 5L, chromosome_length = 100)
p2 <- d_cibd(x = 100, ibd = 0, pedigree = ped_halfsibs(5))

p1
p2
stopifnot(all.equal(p1, p2))

# If you have five children, which fraction of chromosome 1
# is passed down to the next generation? Assume a length of 268 cM

# Pr. of passing down the complete grand-maternal and grand-paternal parts
# is 42%
pr_all_genes_survive(number_of_children = 5L, chromosome_length = 268)

# The distribution of the fraction that is passed down has a point
# mass of 0.42 at 100% and has a continuous density with weight 0.58
dist <- total_ibd_dist(ped_halfsibs(5), ibd_state = 0,
chromosome_length = 267, fraction = TRUE)
```

```
dist

plot(dist)
```

`r_cibd` *Random generation for IBD on a continuous genome*

Description

The `r_cibd` function generates random Identity-by-Descent (IBD) segments along a continuous genome, given a specified pedigree and observed individuals.

Usage

```
r_cibd(
  n,
  pedigree,
  ids = pedtools::leaves(pedigree),
  states = "ibd",
  ibd_state = 1L,
  chromosome_length = 267.77
)
```

Arguments

`n`	Number of observations
`pedigree`	Pedigree in `pedtools::ped` form.
`ids`	Ids for which IBD is observed. Default is `pedtools::leaves(pedigree)`.
`states`	One of `"ibd"` (default), `"kappa"`, `"identity"` or `"detailed"`.
`ibd_state`	Default is 1.
`chromosome_length`	Default is 267.77 cM (an estimate of the length of chromosome 1).

Value

A list containing:

`samples`	Data frame of simulated IBD segments along the chromosome.
`stats`	Data frame with summary statistics per sample, including total IBD length and the segment count.

Examples

```
## Basic example: IBD along one chromosome for half siblings
L <- 300
r_hs <- r_cibd(n = 1e4, pedigree = pedtools::halfSibPed(), chromosome_length = 300)

# half sibs alternate between IBD (state 1) and not IBD (state 0)
head(r_hs$samples)

# the total_length and number of segments per sample are also returned
head(r_hs$stats)

## Comparing half siblings and grandparent-grandchild
r_gp <- r_cibd(n = 1e4, pedigree = pedtools::linearPed(2), ids = c(1, 5),
               chromosome_length = 300)

hist(r_gp$stats$total_length)
hist(r_hs$stats$total_length)
```

---

sd *Standard Deviation for IBD Distribution Objects*

Description

Computes the standard deviation of a total IBD or segment count distribution.

Usage

```
sd(x, ...)
```

Arguments

x	An object of class `total_ibd_dist` or `segment_count_dist`.
...	Additional arguments (not used).

Value

A numeric value.

segment_count_dist *Compute probability distribution of IBD segment count*

Description

The segment_count_dist function computes the probability distribution of the number of IBD segments (i.e., the count of IBD intervals) between pairs of individuals in a pedigree, for a given IBD state and chromosome length(s).

Usage

```
segment_count_dist(
  pedigree,
  ids = pedtools::leaves(pedigree),
  states = "ibd",
  ibd_state = 1L,
  chromosome_length = 267.77,
  convolve = TRUE
)
```

Arguments

pedigree	Pedigree in pedtools::ped form.
ids	Ids for which IBD is observed. Default is pedtools::leaves(pedigree).
states	One of "ibd" (default), "kappa", "identity" or "detailed".
ibd_state	Default is 1.
chromosome_length	Default is 267.77 cM (an estimate of the length of chromosome 1).
convolve	Should the distribution of the sum (across chromosomes) be obtained?

Details

This function is analogous to total_ibd_dist() but focuses on the number of IBD segments rather than the total IBD length.

Value

object of class segment_count_dist

See Also

total_ibd_dist() for the distribution of IBD *length*.

Examples

```
ped_hs <- pedtools::halfSibPed()

dist <- segment_count_dist(ped_hs, chromosome_length = 100)
dist
plot(dist)
E(dist)
sd(dist)

r <- r_cibd(n = 1e4, pedigree = ped_hs, chromosome_length = 100)
mean(r$stats$segment_count)
sd(r$stats$segment_count)
```

`total_ibd_dist` *Compute probability distribution of total IBD*

Description

The `total_ibd_dist` function computes the probability distribution of the total IBD length (or fraction) over one or more autosomes.

Usage

```
total_ibd_dist(
  pedigree,
  ids = pedtools::leaves(pedigree),
  fraction = FALSE,
  states = "ibd",
  ibd_state = 1L,
  chromosome_length = 267.77,
  convolve = TRUE,
  ...
)
```

Arguments

`pedigree`	Pedigree in `pedtools::ped` form.
`ids`	Ids for which IBD is observed. Default is `pedtools::leaves(pedigree)`.
`fraction`	If TRUE, the distribution of the IBD fraction instead of length will be returned. Default is FALSE.
`states`	One of `"ibd"` (default), `"kappa"`, `"identity"` or `"detailed"`.
`ibd_state`	Default is 1.
`chromosome_length`	Default is 267.77 cM (an estimate of the length of chromosome 1).
`convolve`	Should the distribution of the sum (across chromosomes) be obtained?
`...`	Additional parameters passed to `convolve_total_ibd_dists` when `convolve=TRUE`.

Details

For many pedigree relationships, the probability distribution of the total IBD length over one chromosome is a mixture of two point masses (0 and chromosome length) and a continuous density.

If `convolve=TRUE` (the default) and `chromosome_length` has length greater than one, the convolution of the distributions will be obtained by FFT using the `convolve_total_ibd_dists` function. Convolution will typically produce a rapidly increasing number of point masses with very small probabilities which are discarded if the probability falls below a threshold of `1e-9`; see `convolve_total_ibd_dists` for details and finer control.

Value

object of class `total_ibd_dist`

Examples

```
## Total IBD and fraction of IBD for a cousin relationship
ped_fc <- pedtools::cousinPed()

# total IBD length for 100 cM
dist_length <- total_ibd_dist(ped_fc, chromosome_length = 100)
dist_length
plot(dist_length)

# fraction IBD for 100 cM
dist_fraction <- total_ibd_dist(ped_fc, chromosome_length = 100,
                                fraction = TRUE)
dist_fraction
plot(dist_fraction)

# distribution of total length across three chromosomes (150, 200, 250 cM)
plot(total_ibd_dist(ped_fc, chromosome_length = c(150, 200, 250)))

# a quick approximation with reasonable accuracy (with just 256 gridpoints)
plot(total_ibd_dist(ped_fc, chromosome_length = c(150, 200, 250),
                    number_of_gridpoints_exponent = 8))

## Difference between IBD distributions between half-sibs, uncle-nephew
## and grandparent-grandchild relationships

# kappa1 is 1/2 for half sibs, uncle-nephew and grandparent-grandchild
# but the distribution of the fraction of a chromosome that is in this
# state differs between the relationships

# define pedigrees and verify kappa1
ped_gp <- pedtools::linearPed(n = 2)
ped_av <- pedtools::avuncularPed()
ped_hs <- pedtools::halfSibPed()

stopifnot(all.equal(1/2,
          d_ibd(1, ped_av),
          d_ibd(1, ped_hs),
```

```
          d_ibd(1, ped_gp, ids = c(1,5))))

# Compute IBD distributions
d_av <- total_ibd_dist(ped_av)
d_hs <- total_ibd_dist(ped_hs)
d_gp <- total_ibd_dist(ped_gp, ids = c(1,5))

# the point masses are different
d_av
d_hs
d_gp

# plot the continuous densities
x0 <- seq(0, 267.77, length.out = 200)
df <- data.frame(cM = rep(x0, 3),
                 y = c(d(d_av)(x0), d(d_hs)(x0), d(d_gp)(x0)),
                 Relationship = rep(c("Avuncular", "Half-Sibling",
                                      "Grandparent"), each = length(x0)))

require(ggplot2)
ggplot(df, aes(x = cM, y = y, color = Relationship)) + geom_line()
```

`total_ibd_dist_moments` *Compute moments of probability distribution of total IBD*

Description

The `total_ibd_dist_moments` function computes mean and variance of the probability distribution of the total IBD length (or fraction) over one autosome. The function uses double numerical integration.

Usage

```
total_ibd_dist_moments(
  pedigree,
  ids = pedtools::leaves(pedigree),
  fraction = FALSE,
  states = "ibd",
  ibd_state = 1L,
  chromosome_length = 267.77
)
```

Arguments

`pedigree`	Pedigree in `pedtools::ped` form.
`ids`	Ids for which IBD is observed. Default is `pedtools::leaves(pedigree)`.
`fraction`	If TRUE, the distribution of the IBD fraction instead of length will be returned. Default is FALSE.

states	One of "ibd" (default), "kappa", "identity" or "detailed".
ibd_state	Default is 1.
chromosome_length	Default is 267.77 cM (an estimate of the length of chromosome 1).

Value

list

Examples

```
# Full Siblings are double IBD with 25% probability
# we may compute the expectation and variance of the total length of
# a chromosome that is double IBD for a chromosome of 100 cM (i.e. 1 Morgan)
m <- total_ibd_dist_moments(pedigree = pedtools::nuclearPed(nch = 2),
                        ibd_state = 2, chromosome_length = 100)
m

# compare to numerical integration of the full distribution
d_fs <- total_ibd_dist(pedigree = pedtools::nuclearPed(nch = 2),
ibd_state = 2, chromosome_length = 100)

m2 <- list(mean = E(d_fs), variance = var(d_fs), sd = sd(d_fs))
m2

stopifnot(all.equal(m, m2))

# Expectation and variance of _fraction_ of the genome that is
# double IBD between four full siblings
m4 <- total_ibd_dist_moments(pedigree = pedtools::nuclearPed(nch = 4),
ibd_state = 2, chromosome_length = 100, fraction = TRUE)
m4

stopifnot(all.equal(0.25^3, m4$mean))
```

---

var *Variance for IBD Distribution Objects*

Description

Computes the variance of a total IBD or segment count distribution.

Usage

```
var(x, ...)
```

Arguments

x	An object of class total_ibd_dist or segment_count_dist.
...	Additional arguments (not used).

Value

A numeric value.

Index