Package 'ribd'

November 26, 2021

Type Package

Title Pedigree-based Relatedness Coefficients

Version 1.3.1

Description Recursive algorithms for computing various relatedness coefficients, including pairwise kinship, kappa and identity coefficients. Both autosomal and X-linked coefficients are computed. Founders are allowed to be inbred, enabling construction of any given kappa coefficients (Vigeland (2020) <doi:10.1007/s00285-020-01505-x>). In addition to the standard pairwise coefficients, 'ribd' also computes a range of lesser-known coefficients, including generalised kinship coefficients (Karigl (1981) <doi:10.1111/j.1469-1809.1981.tb00341.x>; Weeks and Lange (1988) <https:www.ncbi.nlm.nih.gov/pmc/articles/PMC1715269>), two-locus coefficients (Thompson (1988) <doi:10.1093/imammb/5.4.261>) and multi-person coefficients. This package is part of the 'ped suite', a collection of packages for pedigree analysis in R. Several methods of 'ribd' are featured in the online app 'QuickPed' available at <https://magnusdv.shinyapps.io/quickped>.

License GPL-3

URL https://github.com/magnusdv/ribd, https://magnusdv.github.io/pedsuite/

Depends pedtools, R (>= 3.5.0)

Imports glue, kinship2, slam

Suggests identity, testthat

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.1.2

NeedsCompilation no

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 2021-11-26 16:10:59 UTC

R topics documented:

`condensedIdentity` *Condensed identity coefficients*

Description

Computes the 9 condensed identity coefficients of pairwise relationships in a pedigree. Founders of the pedigree may be inbred; use `pedtools::founderInbreeding()` to set this up.

Usage

```
condensedIdentity(x, ids, sparse = NA, verbose = FALSE, checkAnswer = verbose)
```

Arguments

x	A pedigree in the form of a pedtools::ped object
ids	A character (or coercible to character) containing ID labels of two or more pedigree members.
sparse	A positive integer, indicating the pedigree size limit for using sparse arrays (as implemented by the slam package) instead of ordinary arrays.
verbose	A logical
checkAnswer	A logical. If TRUE, and the identity package is installed, the result is checked against the output of identity::identity.coefs(). This option is ignored if any of the founders are inbred, or if ids has length greater than 2.

Details

The implementation is a modified version of Karigl's recursive algorithm (1981).

Value

If ids has length 2: A vector of length 9, containing the condensed identity coefficients.

If ids has length > 2: A data frame with one row for each pair of individuals, and 11 columns. The first two columns contain the ID labels, and columns 3-11 contain the condensed identity coefficients.

References

G. Karigl (1981). *A recursive algorithm for the calculation of identity coefficients* Annals of Human Genetics, vol. 45.

See Also

kappa(), condensedIdentityX(), pedtools::founderInbreeding()

Examples

```
# One generation of full sib mating.
# (One of the simplest examples with all 9 coefficients nonzero.)
x = fullSibMating(1)
j1 = condensedIdentity(x, ids = 5:6)

stopifnot(all.equal(j1, c(2, 1,4, 1, 4, 1, 7, 10, 2)/32))

# Recalculate the coefficients when the founders are 100% inbred
founderInbreeding(x, 1:2) = 1
condensedIdentity(x, ids = 5:6)
```

`condensedIdentityX` *Identity coefficients on X*

Description

Computes the X chromosomal condensed identity coefficients of a pairwise relationship.

Usage

```
condensedIdentityX(x, ids, sparse = NA, verbose = FALSE)
```

Arguments

`x`	A pedigree in the form of a `pedtools::ped` object
`ids`	A character (or coercible to character) containing ID labels of two or more pedigree members.
`sparse`	A positive integer, indicating the pedigree size limit for using sparse arrays (as implemented by the slam package) instead of ordinary arrays.
`verbose`	A logical

Details

The implementation is inspired by Karigl's recursive algorithm (1981) for the autosomal case, modified to account for X-linked inheritance.

The X chromosomal pairwise identity states depend on the sexes of the two individuals. If both are female, the states are the same as in the autosomal case. When males are involved, the two individuals have less than 4 alleles, hence the states differ from the autosomal ones. However, to avoid drawing (and learning) new pictures we re-use the autosomal states by using the following simple rule: **Replace any hemizygous male allele with a pair of autozygous alleles**. In this way each X state corresponds to a unique autosomal state.

For simplicity the output always contains 9 coefficients, but with NA's in the positions of undefined states (depending on the sex combination). The README file on the GitHub home page of ribd has a table illustrating this.

Value

If `ids` has length 2: A vector of length 9, containing the condensed identity coefficients. If any of the individuals are male, certain states are undefined, and the corresponding coefficients are NA. (See Details.)

If `ids` has length > 2: A data frame with one row for each pair of individuals, and 11 columns. The first two columns contain the ID labels, and columns 3-11 contain the condensed identity coefficients.

See Also

`kinshipX()`, `condensedIdentity()`, `pedtools::founderInbreeding()`

Examples

```
x = fullSibMating(1)
x_sisters = swapSex(x, 5)
x_brothers = swapSex(x, 6)

condensedIdentityX(x, ids = 5:6)
condensedIdentityX(x_sisters, ids = 5:6)
condensedIdentityX(x_brothers, ids = 5:6)
```

`constructPedigree` *Pedigree construction*

Description

Construct a pedigree yielding a prescribed set of IBD coefficients.

Usage

```
constructPedigree(kappa, describe = TRUE, verbose = FALSE)
```

Arguments

`kappa`	A probability vector of length 3; $(kappa0, kappa1, kappa2)$.
`describe`	A logical. If TRUE, a textual description of the resulting relationship is printed.
`verbose`	A logical. If TRUE, various details about the calculations are printed.

Details

The construction follows the method and formulae given in Vigeland (2020).

Value

A ped object containing a pair of double half cousins with inbred founders. (In corner cases the relationship collapses into siblings.)

References

M. D. Vigeland (2020). *Relatedness coefficients in pedigrees with inbred founders*. Journal of mathematical biology. doi: 10.1007/s0028502001505x

Examples

```
# Full siblings
x = constructPedigree(kappa = c(0.25, 0.5, 0.25))
kappaIBD(x, leaves(x))

# A relationship halfway between parent-child and full sibs
kap = c(1/8, 6/8, 1/8)
showInTriangle(kap, label = " (1/8, 1/8)", pos = 4)

y = constructPedigree(kappa = kap)
plot(y)

stopifnot(all.equal(kappaIBD(y, leaves(y)), kap))

# kappa = (0,1,0) does not give a parent-child relationship,
# but half siblings whose shared parent is completely inbred.
z = constructPedigree(kappa = c(0,1,0))
plot(z)
```

---

external_coefs *Relatedness coefficients by other programs*

Description

Wrappers for functions in other packages or external programs, computing various relatedness coefficients.

Usage

```
kinship2_kinship(x, ids = NULL)

kinship2_kinshipX(x, ids = NULL)

kinship2_inbreeding(x)

kinship2_inbreedingX(x)

idcoefs(x, ids)

idcoefs2(x, ids, verbose = FALSE, cleanup = TRUE)
```

Arguments

x	A pedigree, in the form of a pedtools::ped object.
ids	A integer vector of length 2.
verbose	A logical, indicating if messages from IdCoefs should be printed.

cleanup — A logical: If TRUE, the pedfile and sample file created for the IdCoefs run are deleted automatically.

Details

`kinship2_inbreeding()` and `kinship2_kinship()` are wrappers of `kinship2::kinship()` with the parameter `chrtype = "autosome"`.

Similarly ,`kinship2_inbreedingX()` and `kinship2_kinshipX()` wrap `kinship2::kinship()` using `chrtype = "x"`.

`idcoefs()` wraps `identity::identity.coefs()`, which is an R interface for the C program `IdCoefs` written by Mark Abney (2009). The `identity.coefs()` function sometimes causes R to crash, hence I have provided an alternative wrapper, `idcoefs2`, which executes an external call to the original C program `IdCoefs` (version 2.1.1). For this to work, `IdCoefs` must be installed on the computer (see link in the References section below) and the executable placed in a folder included in the PATH variable. The `idcoefs2()` wrapper works by writing the necessary files to disk and calling `IdCoefs` via `system()`.

Value

For `kinship2_inbreeding()` and `kinship2_inbreedingX()`, a named numerical vector with the inbreeding coefficients and ID labels as names.

For `kinship2_kinship()` and `kinship2_kinshipX()`, either a single numeric (if `ids` is a pair of pedigree members) or the whole kinship matrix, with the ID labels as dimnames.

For `jaquard()` and `jaquard2()`, a numerical vector of length 9 (in the standard order of Jacquard's condensed identity coefficients).

Author(s)

Magnus Dehli Vigeland

References

Abney, Mark (2009). *A graphical algorithm for fast computation of identity coefficients and generalized kinship coefficients.* Bioinformatics, 25, 1561-1563. http://home.uchicago.edu/~abney/abney_web/Software.html

See Also

`kinship2::kinship()`, `identity::identity.coefs()`

Examples

```
# A random pedigree with 2 founders and 5 matings
p = pedtools::randomPed(g = 5, founders = 2, seed = 111)

# Check that ribd agrees with kinship2 package
kinship_k2 = kinship2_kinship(p)
kinship_ribd = kinship(p)
stopifnot(identical(kinship_k2, kinship_ribd))
```

```
# Check on X also
kinshipX_k2 = kinship2_kinshipX(p)
kinshipX_ribd = kinshipX(p)
stopifnot(identical(kinshipX_k2, kinshipX_ribd))
```

---

generalisedKinship *Generalised kinship coefficients*

Description

Compute generalised single-locus kinship coefficients, using the algorithm by Weeks & Lange (1988).

Usage

```
generalisedKinship(x, pattern, mem = NULL, verbose = FALSE, debug = FALSE)
```

Arguments

x	A ped object.
pattern	A kinPattern object.
mem	An environment (for internal use).
verbose	A logical.
debug	A logical.

Value

A single probability.

References

Weeks and Lange. *The Affected-Pedigree-Member Method of Linkage Analysis*. Am. J. Hum. Genet. 42:315-326, 1988.

Examples

```
x = nuclearPed(3)
kp = kinPattern(x, list(c(1,1,1)))
generalisedKinship(x, kp)


##### IBD coefficients via generalised kinship ###
#(Clearly not the simplest way; serves as a check)
IBD_from_gk = function(x, ids) {
  fa1 = father(x, ids[1])
  fa2 = father(x, ids[2])
```

```
  mo1 = mother(x, ids[1])
  mo2 = mother(x, ids[2])
  GK = function(...) generalisedKinship(x, list(...))

  k0 = GK(fa1, fa2, mo1, mo2)
  k1 = GK(c(fa1, fa2), mo1, mo2) + GK(c(fa1, mo2), fa2, mo1) +
       GK(c(mo1, fa2), fa1, mo2) + GK(c(mo1, mo2), fa1, fa2)
  k2 = GK(c(fa1, fa2), c(mo1, mo2)) + GK(c(fa1, mo2), c(mo1, fa2))
  c(k0, k1, k2)
}

y1 = nuclearPed(2); ids = 3:4
stopifnot(IBD_from_gk(y1, ids) == kappaIBD(y1, ids))

y2 = quadHalfFirstCousins()
ids = 9:10
stopifnot(IBD_from_gk(y2, ids) == kappaIBD(y2, ids))

#### Triple/quad kinship (compare with karigl)
x = fullSibMating(1)
ids = c(1,5,6)
stopifnot(generalisedKinship(x, list(ids)) == generalisedKinship3(x, ids))
ids = c(1,5,6,5)
stopifnot(generalisedKinship(x, list(ids)) == generalisedKinship4(x, ids))
```

`generalised_karigl` *Karigl's generalised kinship coefficients*

Description

Compute generalised kinship coefficients, as defined by Karigl (1981), involving up to 4 pedigree members. The founders may be inbred; see Examples.

Usage

```
generalisedKinship3(
  x,
  ids,
  sparse = NA,
  chromType = "autosomal",
  verbose = FALSE
)

generalisedKinship4(
  x,
  ids,
  sparse = NA,
  chromType = "autosomal",
```

```
  verbose = FALSE
)

generalisedKinship22(
  x,
  ids,
  sparse = NA,
  chromType = "autosomal",
  verbose = FALSE
)
```

Arguments

x	A pedigree, in the form of a pedtools::ped object.
ids	A vector of ID labels, of length 3 for generalisedKinship3() and 4 for generalisedKinship4() and generalisedKinship22().
sparse	A positive integer, indicating the pedigree size limit for using sparse arrays. If NA, a default limit of 50 is used.
chromType	Either "autosomal" or "x".
verbose	A logical.

Details

The function generalisedKinship3() computes the generalised kinship coefficient of three (not necessarily distinct) members a, b and c, defined as the probability that if a random allele is chosen from each of them, they are all identical by descent.

The function generalisedKinship4() computes the generalised kinship coefficient of four individuals, defined similarly to the above.

The function generalisedKinship22() computes the generalised kinship coefficient of two pairs of members, defined as the probability that in both pairs simultaneously, random alleles chosen from the two individuals are IBD.

Value

A numeric of length 1.

See Also

kinship(), kinshipX(), condensedIdentity(), condensedIdentityX()

Examples

```
# Generalised kinship between three siblings
x = nuclearPed(3)
phi3 = generalisedKinship3(x, ids = 3:5)

# Recalculate if the father is 100% inbred
founderInbreeding(x, 1) = 1
```

```
phi3_inbred = generalisedKinship3(x, ids = 3:5)

stopifnot(phi3 == 1/16, phi3_inbred == 1/8 + 1/32)
```

---

ibdDraw *Colourised IBD plot*

Description

This is a pedagogical tools for illustrating the concept of identity-by-descent, by representing the alleles in a pedigree by coloured points or letters. By default, the alleles are placed below each pedigree symbols, but any positions are possible, including inside. (See examples.)

Usage

```
ibdDraw(
  x,
  alleles,
  symbol = c("point", "text"),
  pos = 1,
  cols = NULL,
  cex = NA,
  sep = NULL,
  dist = 1,
  labs = FALSE,
  checkFounders = TRUE,
  checkParents = TRUE,
  margin = c(1, 1, 1, 1),
  ...
)
```

Arguments

x	A ped object.
alleles	A list of length pedsize(x). Each element should consist of one or two integers, representing different colours. Zeroes produce "greyed-out" alleles.
symbol	Either "point" or "text".
pos	A vector recycled to the length of labels(x), indicating allele placement relative to the pedigree symbols: 0 = inside; 1 = below; 2 = left; 3 = above; 4 = right. By default, all are placed below.
cols	A colour vector corresponding to the integers occurring in alleles.
cex	An expansion factor for the allele points/letters. Default: 3 for points and 2 for text.
sep	The separation between haplotypes within a pair, given as a multiple of the width of a pedigree symbol. Default: 0.5 when pos = 0 and 1 otherwise.

dist	The distance between pedigree symbols and the alleles, given as a multiple of the height of a pedigree symbol. Default: 1. Ignored when pos = 0.
labs	A logical indicating if labels should be included.
checkFounders	A logical. If TRUE (default), a warning is issued if a founder has two equal alleles other than 0.
checkParents	A logical. If TRUE (default), a warning is issued if someone's alleles don't match those of the parents. This a superficial test and does not catch all Mendelian errors.
margin	Plot margins (bottom, left, top, right).
...	Further arguments passed on to plot.ped().

Value

The plot structure is returned invisibly.

See Also

pedtools::plot.ped(), ibdsim2::haploDraw()

Examples

```
op = par(no.readonly = TRUE)

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

x = nuclearPed(2)
als = list(1:2, 3:4, c(1,3), c(2,3))

# Default options
ibdDraw(x, als)

# Nicer colors
cols = c(7, 3, 2, 4)
ibdDraw(x, als, cols = cols)

# Inside the pedigree symbols
ibdDraw(x, als, cols = cols, pos = 0, symbolsize = 2.5)

# Other placements (margins depend on device - may need adjustment)
ibdDraw(x, als, cols = cols, pos = c(2, 4, 1, 1),
        margin = c(2, 6, 2, 6))

# Letters instead of points
ibdDraw(x, als, cols = cols, symbol = "text", cex = 2)

# Further arguments (note that `col` is an argument of `ped.plot()`)
ibdDraw(x, als, cols = cols, pos = 0, symbolsize = 2,
        labs = TRUE, hatched = 3:4, col = "blue")
```

```
# Mutations are warned about (unless `checkParents = FALSE`)
ibdDraw(x, alleles = list(1:2, 3:4, 5, 6))


##############################
# Example 2: Cousin pedigree #
##############################

x = swapSex(cousinPed(1), 3)
als = list(1:2, 3:4, NULL, c(1,3), c(2,3), NULL, 3, 3)

cols = c(7, 3, 2, 4)
ibdDraw(x, als, cols = cols, dist = 0.8)
ibdDraw(x, als, cols = cols, dist = 0.8, symbol = "text")

# Alternative: 0's give greyed-out alleles
als2 = list(1:2, 3:4, c(0,0), c(1,3), c(2,3), c(0,0), c(0,3), c(3,0))

ibdDraw(x, als2, cols = cols, dist = 0.8)
ibdDraw(x, als2, cols = cols, dist = 0.8, symbol = "text")


############################
# Example 3: X inheritance #
############################

x = nuclearPed(2, sex = c(1, 2))
als = list(1, 2:3, 3, c(1, 3))
ibdDraw(x, als, cols = c(3, 7, 2))


#################################
# Example 4: mtDNA inheritance  #
#################################

x = linearPed(2, sex = 2)
als = list(1, 2, 2, 3, 2)
ibdDraw(x, als, cols = 2:4)


# Restore graphics parameters
par(op)
```

---

ibdTriangle *IBD triangle plot*

Description

The IBD triangle is typically used to visualize the pairwise relatedness of non-inbred individuals. Various annotations are available, including points marking the most common relationships, contour lines for the kinship coefficients, and shading of the unattainable region.

Usage

```
ibdTriangle(
  relationships = c("UN", "PO", "MZ", "S", "H,U,G", "FC"),
  pch = 16,
  cexPoint = 1.2,
  cexText = 1.2,
  kinshipLines = numeric(),
  shading = "lightgray",
  xlim = c(0, 1),
  ylim = c(0, 1),
  axes = FALSE,
  xlab = expression(kappa[0]),
  ylab = expression(kappa[2]),
  cexLab = cexText,
  mar = c(3.1, 3.1, 1, 1),
  xpd = TRUE,
  keep.par = TRUE
)
```

Arguments

relationships	A character vector indicating relationships points to be included in the plot. See Details for a list of valid entries.
pch	Symbol used for the relationship points (see par()).
cexPoint	A number controlling the symbol size for the relationship points.
cexText	A number controlling the font size for the relationship labels.
kinshipLines	A numeric vector (see Details).
shading	The shading colour for the unattainable region.
xlim, ylim, mar, xpd	Graphical parameters; see par().
axes	A logical: Draw surrounding axis box? Default: FALSE.
xlab, ylab	Axis labels.
cexLab	A number controlling the font size for the axis labels.
keep.par	A logical. If TRUE, the graphical parameters are not reset after plotting, which may be useful for adding additional annotation.

Details

For any pair of non-inbred individuals A and B, their genetic relationship can be summarized by the IBD coefficients $(\kappa_0, \kappa_1, \kappa_2)$, where κ_i = P(A and B share i alleles IBD at random autosomal locus).

Since $\kappa_0 + \kappa_1 + \kappa_2 = 1$, any relationship corresponds to a point in the triangle in the (κ_0, κ_2)-plane defined by $\kappa_0 \geq 0, \kappa_2 \geq 0, \kappa_0 + \kappa_2 \leq 1$. The choice of κ_0 and κ_2 as the axis variables is done for reasons of symmetry and is not significant (other authors have used different views of the triangle).

As shown by Thompson (1976), points in the subset of the triangle defined by $4\kappa_0\kappa_2 > \kappa_1^2$ are unattainable for pairwise relationships. By default this region in shaded in a 'light grey' colour, but this can be modified with the `shading` argument.

The IBD coefficients are linearly related to the kinship coefficient ϕ by the formula

$$\phi = 0.25\kappa_1 + 0.5\kappa_2.$$

By indicating values for ϕ in the `kinshipLines` argument, the corresponding contour lines are shown as dashed lines in the triangle plot.

The following abbreviations are valid entries in the `relationships` argument:

- UN = unrelated
- PO = parent/offspring
- MZ = monozygotic twins
- S = full siblings
- H,U,G = half sibling/avuncular (**u**ncle)/grandparent
- FC = first cousins
- SC = second cousins
- DFC = double first cousins
- Q = quadruple first half cousins

Value

None

Author(s)

Magnus Dehli Vigeland

References

- E. A. Thompson (1975). *The estimation of pairwise relationships.* Annals of Human Genetics 39.
- E. A. Thompson (1976). *A restriction on the space of genetic relationships.* Annals of Human Genetics 40.

Examples

```
opar = par(no.readonly = TRUE) # store graphical parameters

ibdTriangle()
ibdTriangle(kinshipLines = c(0.25, 0.125), shading = NULL, cexText = 0.8)

par(opar) # reset graphical parameters
```

`inbreeding` *Inbreeding coefficients*

Description

Compute the inbreeding coefficients of all members of a pedigree. These are simple wrappers of `kinship()` and `kinshipX()`. The founders may be inbred; see `pedtools::founderInbreeding()` for how to set this up.

Usage

```
inbreeding(x, ids = NULL, id = NULL)

inbreedingX(x, ids = NULL, id = NULL)
```

Arguments

`x`	A pedigree, in the form of a `pedtools::ped` object.
`ids`	A vector of ID labels, or NULL (default).
`id`	Deprecated; use `ids` instead.

Details

The autosomal inbreeding coefficient of a pedigree member is defined as the probability that, at a random autosomal locus, the two alleles carried by the member are identical by descent relative to the pedigree. It follows from the definition that the inbreeding coefficient of a member equals the kinship coefficient of the parents.

The X chromosomal inbreeding coefficient of an female member is defined similarly to the autosomal case above. For males is it always 1.

The inbreeding coefficients are computed from the diagonal of the kinship matrix, by the formula

$$f_a = 2 * \phi_{aa} - 1.$$

Value

If `ids` has length 1, the inbreeding coefficient of this individual is returned as a single unnamed number.

Otherwise, the output is a named numeric vector containing the inbreeding coefficients of the indicated pedigree members (if `ids = NULL`: all).

See Also

`kinship()`

Examples

```
# Child of half siblings: f = 1/8
x = halfCousinPed(0, child = TRUE)
inbreeding(x)

# If the father is 100% inbred, the inbreeding coeff of the child doubles
fa = commonAncestors(x, 4:5) # robust to label change
founderInbreeding(x, fa) = 1

inbreeding(x)

# Simpler output using the `ids` argument:
inbreeding(x, ids = 6)

### X-chromosomal inbreeding coefficients ###
# These depend on the genders in the pedigree.
# To exemplify, we consider a child of half siblings.

xPat = halfSibPed(sex2 = 2) # paternal half sibs
xPat = addChildren(xPat, father = 4, mother = 5, nch = 1, sex = 2)
stopifnot(inbreedingX(xPat, ids = 6) == 0)

# Change to maternal half sibs => coeff becomes 1/4.
xMat = swapSex(xPat, 1)
stopifnot(inbreedingX(xMat, ids = 6) == 0.25)

# Example with selfing and complete inbreeding
s = selfingPed(1)
founderInbreeding(s, 1) = 1
inbreeding(s, ids = 2)
```

---

jicaque *Jicaque pedigree*

Description

A data frame describing a pedigree from the Jicaque tribe, studied by Chapman and Jacquard (1971).

Usage

```
jicaque
```

Format

A data frame with 22 rows and four columns:

- `id` : individual ID
- `fid` : father's ID (or 0 if not included)

- `mid` : mother's ID (or 0 if not included)
- `sex` : Gender codes, where 1 = male and 2 = female

References

Chapman, A.M and Jacquard, A. (1971). Un isolat d'Amerique Centrale: les Indiens Jicaques de Honduras. In Genetique et Population. Paris: Presses Universitaires de France.

`kappaIBD` *IBD (kappa) coefficients*

Description

Computes the three IBD coefficients summarising the relationship between two non-inbred individuals. Both autosomal and X chromosomal versions are implemented.

Usage

```
kappaIBD(x, ids = labels(x), inbredAction = 1, simplify = TRUE)

kappaIbdX(x, ids, sparse = NA, verbose = FALSE)
```

Arguments

`x`	A pedigree in the form of a `ped` object (or a list of such).
`ids`	A character (or coercible to character) containing ID labels of two or more pedigree members.
`inbredAction`	An integer telling the program what to do if either of the `ids` individuals are inbred. Possible values are: 0 = do nothing; 1 = print a warning message (default); 2 = raise an error. In the first two cases the coefficients are reported as `NA`.
`simplify`	Simplify the output (to a numeric of length 3) if `ids` has length 2. Default: TRUE.
`sparse`	A positive integer, indicating the pedigree size limit for using sparse arrays (as implemented by the slam package) instead of ordinary arrays.
`verbose`	A logical.

Details

For non-inbred individuals a and b, their autosomal IBD coefficients $(\kappa 0, \kappa 1, \kappa 2)$ are defined as follows:

$$\kappa_i = P(aandbshareiallelesIBDatarandomautosomallocus)$$

The autosomal kappa coefficients are computed from the kinship coefficients. When a and b are both nonfounders, the following formulas are well-known:

- $\kappa 2 = \phi_M M * \phi_F F + \phi_M F * \phi_F M$

- $\kappa 1 = 4 * \phi_a b - 2 * \kappa 2$
- $\kappa 0 = 1 - \kappa 1 - \kappa 2$

Here $\phi_M M$ denotes the kinship coefficient between the mothers of a and b, and so on. If either a or b is a founder, then $\kappa 2 = 0$, while the other two formulas remain as above.

The X chromosomal IBD coefficients are defined as in the autosomal case, with the exception that $\kappa 2$ is undefined when at least one of the two individuals is male. Hence the computation is greatly simplified when males are involved. Denoting the standard kinship coefficient by ϕ, the formulas are:

- Both male: $(\kappa 0, \kappa 1, \kappa 2) = (1 - \phi, \phi, NA)$
- One male, one female: $(\kappa 0, \kappa 1, \kappa 2) = (1 - 2 * \phi, 2 * \phi, NA)$
- Two females: As in the autosomal case.

Value

If `ids` has length 2 and `simplify = TRUE`: A numeric vector of length 3: $(\kappa 0, \kappa 1, \kappa 2)$.

Otherwise: A data frame with one row for each pair of individuals, and 5 columns. The first two columns contain the ID labels, and columns 3-5 contain the IBD coefficients.

Unless `inbredAction = 2`, the coefficients of pairs involving inbred individuals (inbred *females* in the X version) are reported as NA. Furthermore, the X chromosomal $\kappa 2$ is NA whenever at least one of the two individuals is male.

See Also

`kinship()`, `condensedIdentity()`

Examples

```
### Siblings
x = nuclearPed(2)
k = kappaIBD(x, 3:4)
stopifnot(identical(k, c(.25, .5, .25)))

### Quad half first cousins
x = quadHalfFirstCousins()
k = kappaIBD(x, leaves(x))
stopifnot(identical(k, c(17/32, 14/32, 1/32)))

### Paternal half brothers with 100% inbred father
# Genetically indistinguishable from an (outbred) father-son relationship
x = halfSibPed()
ids = 4:5

# Set founder inbreeding
fou = commonAncestors(x, ids) # robust to label change
founderInbreeding(x, fou) = 1

k = kappaIBD(x, ids)
stopifnot(identical(k, c(0, 1, 0)))
```

kinPattern *Generalised kinship pattern*

Description

Generalised kinship pattern

Usage

```
kinPattern(x, pattern, internal = FALSE)
```

Arguments

x	A ped object
pattern	A list of vectors of ID labels.
internal	A logical

Value

An object of class kinPattern.

Examples

```
kinPattern(nuclearPed(2), list(1, 3:4))
```

kinship *Kinship coefficients*

Description

Compute the matrix of kinship coefficients (autosomal or X) of all members of a pedigree. The founders may be inbred; see pedtools::founderInbreeding() for how to set this up.

Usage

```
kinship(x, ids = NULL)

kinshipX(x, ids = NULL)
```

Arguments

x	A ped object or a list of such.
ids	Either a character of length 2, or NULL. In the former case, it must contain the ID labels of two members of x, and the function will return their kinship coefficient as a single number. If ids is NULL (this is the default), the output is the complete kinship matrix.

Details

For two (not necessarily distinct) members A, B of a pedigree, their autosomal (resp. X) *kinship coefficient* is defined as the probability that random alleles sampled from A and B at the same autosomal (resp. X) locus, are identical by descent relative to the pedigree.

Value

If `ids = NULL`, a symmetric matrix containing all pairwise kinship coefficients in `x`. If `ids` has length 2, the function returns a single number.

See Also

`inbreeding()`, `kappa()`

Examples

```
# Kinship coefficients in a nuclear family with two children
x = nuclearPed(2)
kinship(x)

# X chromosomal kinship coefficients in the same family
kinshipX(x)

# Recalculate the autosomal kinships if the father is 100% inbred
founderInbreeding(x, 1) = 1
kinship(x)
```

`minimalPattern` *Minimal IBD pattern*

Description

Compute the minimal form of given multiperson IBD pattern.

Usage

```
minimalPattern(x)
```

Arguments

`x`	An integer vector of even length.

Value

An integer vector of the same length as `x`.

Examples

```
v = c(1,2,2,3)
stopifnot(identical(minimalPattern(v), c(1,2,1,3)))
```

`multiPersonIBD` *Multi-person IBD coefficients*

Description

Computes the probabilities (coefficients) of all possible patterns of identity by descent (IBD) sharing at a single locus, among N>1 non-inbred members of a pedigree. The reported coefficients are "condensed" in the sense that allele ordering within each individual is ignored. For N = 2, the result should agree with the traditional "kappa" coefficients, as computed by `kappaIBD()`. This function is under development, and should be regarded as experimental. For now, the only cases handled are those with: N = 2 or 3, autosomal locus.

Usage

```
multiPersonIBD(x, ids, complete = FALSE, verbose = FALSE)
```

Arguments

`x`	A ped object.
`ids`	A vector of ID labels.
`complete`	A logical. If FALSE, only IBD patterns with nonzero probability are included in the output.
`verbose`	A logical. If TRUE, some computational details are printed.

Details

Consider N members of a pedigree, i1, i2, ... iN. A pattern of IBD sharing between these individuals is a sequence of N ordered pairs of labels, (a1_1, a1_2), (a2_1, a2_2), ... (aN_1, aN_2), where ai_1 and ai_2 represent the paternal and maternal allele of individual i, respectively. Equality of labels means that the corresponding alleles are IBD, and vice versa.

We say that two IBD patterns are equivalent if one can be transformed into the other by some combination of

- renaming the labels (without changing the structure)
- swapping the paternal/maternal labels of some individuals

Each equivalence class has a "minimal" element, using integer labels, and being minimal with respect to standard sorting. For example, the minimal element equivalent to (a,c),(d,c),(b,b) is (1,2),(2,3),(4,4).

Value

A data frame in which each row corresponds to an equivalence class of multi-person IBD patterns. The first column gives the calculated probability, followed by one column for each `ids` individual, describing the minimal element of the equivalence class. (See Details.) If `complete = FALSE` (the default) rows with probability 0 are removed.

Examples

```
### Trivial example: Trio ###
x = nuclearPed(1)
ids = 1:3
multiPersonIBD(x, ids, complete = TRUE)

### Example due to Peter Green ###
# Three (pariwise) cousins arranged in two different ways,
# with different 3-way IBD coefficients.

threeCousins1 = ped(
  id  = c('gf','gm','gf1','gf2','gf3','gm1','gm2','gm3',
          'f1','f2','f3','m1','m2','m3','c1','c2','c3'),
  fid = c(0,0,0,0,0,0,0,0,'gf1','gf2','gf3','gf','gf','gf',
          'f1','f2','f3'),
  mid = c(0,0,0,0,0,0,0,0,'gm1','gm2','gm3','gm','gm','gm',
          'm1','m2','m3'),
  sex = c(1,2,1,1,1,2,2,2,1,1,1,2,2,2,1,1,1))

threeCousins2 = ped(
  id  = c('gf1','gf2','gf3','gm1','gm2','gm3','f1','f2','f3',
          'm1','m2','m3','c1','c2','c3'),
  fid = c(0,0,0,0,0,0,'gf2','gf3','gf1','gf3','gf1','gf2',
          'f1','f2','f3'),
  mid = c(0,0,0,0,0,0,'gm2','gm3','gm1','gm3','gm1','gm2',
          'm1','m2','m3'),
  sex = c(1,1,1,2,2,2,1,1,1,2,2,2,1,1,1))

ids = c('c1','c2','c3')
multiPersonIBD(threeCousins1, ids)
multiPersonIBD(threeCousins2, ids)
```

`ribd` *ribd: Computation of pedigree-based relatedness coefficients*

Description

Recursive algorithms for computing various relatedness coefficients, including Jacquard's condensed identity coefficients. The standard algorithms are extended to allow inbred founders. Both autosomal and X-linked coefficients are computed.

`showInTriangle` *Add points to the IBD triangle*

Description

Utility function for plotting points in the IBD triangle.

Usage

```
showInTriangle(
  kappa,
  new = TRUE,
  col = 6,
  cex = 1,
  pch = 4,
  lwd = 2,
  labels = FALSE,
  colLab = col,
  cexLab = 0.8,
  pos = 1,
  adj = NULL,
  keep.par = TRUE,
  ...
)
```

Arguments

`kappa`	Coordinates of points to be plotted in the IBD triangle. Valid input types are: • A numerical vector of length 2 or 3. In the latter case `kappa[c(1,3)]` is used. • A matrix of data frame, whose column names must include either `k0` and `k2`, `kappa0` and `kappa2`, or `ibd0` and `ibd2`. • A list (and not a data frame), in which case an attempt is made to bind the elements row-wise.
`new`	A logical indicating if a new triangle should be drawn.
`col, cex, pch, lwd`	Parameters passed onto `points()`.
`labels`	A character of same length as the number of points, or a single logical `TRUE` or `FALSE`. If `TRUE`, an attempt is made to create labels by pasting columns `ID1` and `ID2` in `kappa`, if these exist. By default, no labels are plotted.
`colLab, cexLab, pos, adj`	Parameters passed onto `text()` (if `labels` is non-NULL).
`keep.par`	A logical. If TRUE, the graphical parameters are not reset after plotting, which may be useful for adding additional annotation.
`...`	Plot arguments passed on to `ibdTriangle()`.

Value

None

Author(s)

Magnus Dehli Vigeland

Examples

```
showInTriangle(c(3/8, 1/8), label = "3/4 siblings", pos = 1)
```

`twoLocusIBD` *Two-locus IBD coefficients*

Description

Computes the 3*3 matrix of two-locus IBD coefficients of a pair of non-inbred pedigree members, for a given recombination rate.

Usage

```
twoLocusIBD(
  x,
  ids,
  rho,
  coefs = NULL,
  detailed = FALSE,
  uniMethod = 1,
  verbose = FALSE
)
```

Arguments

`x`	A pedigree in the form of a `pedtools::ped` object.
`ids`	A character (or coercible to character) containing ID labels of two pedigree members.
`rho`	A number in the interval $[0, 0.5]$; the recombination rate between the two loci.
`coefs`	A character indicating which coefficient(s) to compute. A subset of `c('k00','k01','k02','k10','k11'` By default, all coefficients are computed.
`detailed`	A logical, indicating whether the condensed (default) or detailed coefficients should be returned.
`uniMethod`	Either 1 or 2 (for testing purposes)
`verbose`	A logical.

Details

Let A, B be two pedigree members, and L1, L2 two loci with a given recombination rate ρ. The two-locus IBD coefficients $\kappa_{i,j}(\rho)$, for $0 \leq i, j \leq 2$ are defined as the probability that A and B have `i` alleles IBD at L1 and `j` alleles IBD at L2 simultaneously. Note that IBD alleles at the two loci are not required to be *in cis* (or *in trans* for that matter).

The method of computation depends on the (single-locus) IBD coefficient κ_2. If this is zero (e.g. if A is a direct ancestor of B, or vice versa) the two-locus IBD coefficients are easily computable from the two-locus kinship coefficients, as implemented in `twoLocusKinship()`. In the general case, the computation is more involved, requiring *generalised two-locus kinship* coefficients. This is implemented in the function `twoLocusGeneralisedKinship()`, which is not exported yet.

Value

By default, a symmetric 3*3 matrix containing the two-locus IBD coefficients $\kappa_{i,j}$.

If either `coefs` is explicitly given (i.e., not NULL), or `detailed = TRUE`, the computed coefficients are returned as a named vector.

See Also

`twoLocusKinship()`

Examples

```
# Some variables used in several examples below
rseq = seq(0, 0.5, length = 11)  # recombination values

xlab = "Recombination rate"
main = expression(paste("Two-locus IBD:  ", kappa[`1,1`]))

######################################################################
# Example 1: A classic example of three relationships with the same
# one-locus IBD coefficients, but different two-locus coefficients.
# As a consequence, these relationships cannot be separated using
# unlinked markers, but are (theoretically) separable with linked
# markers.
######################################################################
peds = list(
    GrandParent = list(ped = linearPed(2),    ids = c(1, 5)),
    HalfSib     = list(ped = halfSibPed(),    ids = c(4, 5)),
    Uncle       = list(ped = cousinPed(0, 1), ids = c(3, 6)))

# Compute `k11` for each rho
kvals = sapply(peds, function(x)
  sapply(rseq, function(r) twoLocusIBD(x$ped, x$ids, r, coefs = "k11")))

# Plot
matplot(rseq, kvals, type = "l", xlab = xlab, ylab = "", main = main)
legend("topright", names(peds), col = 1:3, lty = 1:3)
```

```
############################################################
# Example 2: Inspired by Fig. 3 in Thompson (1988),
# and its erratum: https://doi.org/10.1093/imammb/6.1.1.
#
# These relationships are also analysed in ?twoLocusKinship,
# where we show that they have identical two-locus kinship
# coefficients. Here we demonstrate that they have different
# two-locus IBD coefficients.
############################################################

# List of pedigrees and ID pairs
GG = linearPed(3)
HU = halfCousinPed(0, removal = 1)
peds = list(
  GreatGrand = list(ped = GG, ids = c(1, 7)),
  HalfUncle  = list(ped = HU, ids = leaves(HU))
)

# Compute `k11` for each rho
kvals = sapply(peds, function(x)
  sapply(rseq, function(r) twoLocusIBD(x$ped, x$ids, r, coefs = "k11")))

# Plot
matplot(rseq, kvals, type = "l", xlab = xlab, ylab = "", main = main)
legend("topright", names(peds), col = 1:2, lty = 1:2)


#####################################################################
# Example 3: Two-locus IBD of two half sisters whose mother have
# inbreeding coefficient 1/4. We compare two different realisations
# of this:
#   PO: the mother is the child of parent-offspring
#  SIB: the mother is the child of full siblings
#
# We show below that these relationships have different two-locus
# coefficients. This exemplifies that a single-locus inbreeding
# coefficient cannot replace the genealogy in analyses of linked loci.
#####################################################################

po = addChildren(nuclearPed(1, sex = 2), 1, 3, nch = 1, sex = 2)
po = addDaughter(addDaughter(po, 4), 4)

sib = addChildren(nuclearPed(2, sex = 1:2), 3, 4, nch = 1)
sib = addDaughter(addDaughter(sib, 5), 5)

plotPedList(list(po, sib), new = TRUE, title = c("PO", "SIB"))

# List of pedigrees and ID pairs
peds = list(PO =  list(ped = po,  ids = leaves(po)),
            SIB = list(ped = sib, ids = leaves(sib)))

# Compute `k11` for each rho
kvals = sapply(peds, function(x)
```

```
  sapply(rseq, function(r) twoLocusIBD(x$ped, x$ids, r, coefs = "k11")))

# Plot
dev.off()
matplot(rseq, kvals, type = "l", xlab = xlab, ylab = "", main = main)
legend("topright", names(peds), col = 1:2, lty = 1:2)

# Check against exact formula
r = rseq
k11_PO = 1/8*(-4*r^5 + 12*r^4 - 16*r^3 + 16*r^2 - 9*r + 5)
stopifnot(all.equal(kvals[, "PO"], k11_PO, check.names = FALSE))

k11_S = 1/16*(8*r^6 - 32*r^5 + 58*r^4 - 58*r^3 + 43*r^2 - 20*r + 10)
stopifnot(all.equal(kvals[, "SIB"], k11_S, check.names = FALSE))


################################################
# Example 4:
# The complete two-locus IBD matrix of full sibs
################################################

x = nuclearPed(2)
k2_mat = twoLocusIBD(x, ids = 3:4, rho = 0.25)
k2_mat

# Compare with explicit formulas
IBDSibs = function(rho) {
  R = rho^2 + (1-rho)^2
  nms = c("ibd0", "ibd1", "ibd2")
  m = matrix(0, nrow = 3, ncol = 3, dimnames = list(nms, nms))
  m[1,1] = m[3,3] = 0.25 *R^2
  m[2,1] = m[1,2] = 0.5 * R * (1-R)
  m[3,1] = m[1,3] = 0.25 * (1-R)^2
  m[2,2] = 0.5 * (1 - 2 * R * (1-R))
  m[3,2] = m[2,3] = 0.5 * R * (1-R)
  m
}

stopifnot(all.equal(k2_mat, IBDSibs(0.25)))


#####################################################
# Example 5: Two-locus IBD of quad half first cousins
#
# We use this to exemplify two simple properties of
# the two-locus IBD matrix.
#####################################################

x = quadHalfFirstCousins()
ids = leaves(x)

# First compute the one-locus IBD coefficients (= c(17, 14, 1)/32)
k1 = kappaIBD(x, ids)
```

```
### Case 1: Complete linkage (`rho = 0`).
# In this case the two-locus IBD matrix has `k1` on the diagonal,
# and 0's everywhere else.
k2_mat_0 = twoLocusIBD(x, ids = ids, rho = 0)

stopifnot(all.equal(k2_mat_0, diag(k1), check.attributes = FALSE))

#' ### Case 2: Unlinked loci (`rho = 0.5`).
# In this case the two-locus IBD matrix is the outer product of
# `k1` with itself.
k2_mat_0.5 = twoLocusIBD(x, ids = ids, rho = 0.5)
stopifnot(all.equal(k2_mat_0.5, k1 %o% k1, check.attributes = FALSE))


########################################################
# Example 6: By Donnelly (1983) these relationships are
# genetically indistinguishable
########################################################

x1 = halfCousinPed(1)
x2 = halfCousinPed(0, removal = 2)
stopifnot(identical(
  twoLocusIBD(x1, ids = leaves(x1), rho = 0.25),
  twoLocusIBD(x2, ids = leaves(x2), rho = 0.25)))
```

`twoLocusIdentity` *Two-locus identity coefficients*

Description

Computes the 9*9 matrix of two-locus condensed identity coefficients of a pair of pedigree members, for a given recombination rate.

Usage

```
twoLocusIdentity(x, ids, rho, coefs = NULL, detailed = FALSE, verbose = FALSE)
```

Arguments

`x`	A pedigree in the form of a `pedtools::ped` object.
`ids`	A character (or coercible to character) containing ID labels of two pedigree members.
`rho`	A number in the interval $[0, 0.5]$; the recombination rate between the two loci.
`coefs`	A character indicating which coefficient(s) to compute. A subset of `c('d00','d01',...,'d99')`. By default, all coefficients are computed.

`detailed`	A logical, indicating whether the condensed (default) or detailed coefficients should be returned.
`verbose`	A logical.

Details

Let A, B be two pedigree members, and L1, L2 two loci with a given recombination rate ρ. The two-locus identity coefficients $\Delta_{i,j}(\rho)$, for $1 \leq i, j \leq 9$ are defined as the probability that the identity state of the alleles of A and B are Σ_i at L1 and Σ_j at L2 simultaneously. (The ordering of the 9 states follows Jacquard (1974).)

For details about the algorithm, see Vigeland (2019).

Value

By default, a symmetric 9*9 matrix containing the two-locus condensed identity coefficients $\Delta_{i,j}$.

If either `coefs` is explicitly given (i.e., not NULL), or `detailed = TRUE`, the computed coefficients are returned as a named vector.

References

M. D. Vigeland (2019) *A recursive algorithm for two-locus identity coefficients* (In progress)

See Also

`twoLocusIBD()`

Examples

```
### Full sibs ###
x = nuclearPed(2)
kapp = twoLocusIBD(x, ids = 3:4, rho = 0.25)
jacq = twoLocusIdentity(x, ids = 3:4, rho = 0.25)
stopifnot(all.equal(jacq[9:7,9:7], kapp, check.attributes = FALSE))

#' ### Parent-child ###
x = nuclearPed(1)
jacq = twoLocusIdentity(x, ids = c(1,3), rho = 0.25)
stopifnot(jacq[8,8] == 1)

### Full sib mating ###
x = fullSibMating(1)
j = condensedIdentity(x, ids = 5:6)
j2 = twoLocusIdentity(x, ids = 5:6, rho = 0.25)
stopifnot(identical(unname(rowSums(j2)), j))
```

`twoLocusKinship` *Two-locus kinship coefficients*

Description

Computes the two-locus kinship coefficient of a pair of pedigree members, at a given recombination rate.

Usage

```
twoLocusKinship(
  x,
  ids,
  rho,
  recombinants = NULL,
  verbose = FALSE,
  debug = FALSE
)
```

Arguments

`x`	A pedigree in the form of a `pedtools::ped` object.
`ids`	A character (or coercible to character) containing ID labels of two or more pedigree members.
`rho`	A numeric vector of recombination rates; all entries must be in the interval $[0, 0.5]$.
`recombinants`	A logical of length 2, applicable only when `ids` has length 2. When given, it indicates whether each of the two gametes is a recombinant or non-recombinant. This parameter is mainly used by `twoLocusIBD()`.
`verbose`	A logical.
`debug`	A logical. If TRUE, detailed messages are printed during the recursion process.

Details

Let A, B be two pedigree members, and L1, L2 two loci with a given recombination rate rho. The two-locus kinship coefficient $\phi_{AB}(rho)$ is defined as the probability that random gametes segregating from A and B has IBD alleles at both L1 and L2 simultaneously.

The implementation is based on the recursive algorithm described by Thompson (1988).

References

E. A. Thompson (1988). *Two-locus and Three-locus Gene Identity by Descent in Pedigrees*. IMA Journal of Mathematics Applied in Medicine & Biology, vol. 5.

Examples

```
######################
# Example 1: Full sibs
######################
x = nuclearPed(2)

k_0 = twoLocusKinship(x, ids = 3:4, rho = 0)
k_0.5 = twoLocusKinship(x, ids = 3:4, rho = 0.5)

stopifnot(k_0 == 1/4, k_0.5 == 1/16)


################################################
# Example 2: Reproducing Fig. 3 in Thompson (1988)
# Note that in the article, curve (a) is wrong.
# See Erratum: https://doi.org/10.1093/imammb/6.1.1
################################################

# Pedigrees (a) - (d)
ped.a = linearPed(3)
ped.b = halfCousinPed(0, removal = 1)
ped.c = cousinPed(1)
ped.d = doubleCousins(1, 1, half1 = TRUE, half2 = TRUE)

peds = list(
  a = list(ped = ped.a, ids = c(1,7)),
  b = list(ped = ped.b, ids = leaves(ped.b)),
  c = list(ped = ped.c, ids = leaves(ped.c)),
  d = list(ped = ped.d, ids = leaves(ped.d))
)

# Recombination values
rseq = seq(0, 0.5, length = 20)

# Compute two-locus kinship coefficients
kvals = sapply(peds, function(x) twoLocusKinship(x$ped, x$ids, rseq))

# Plot
matplot(rseq, kvals, type = "l", lwd = 2)
legend("topright", names(peds), col = 1:4, lty = 1:4, lwd = 2)
```

`twoLocusPlot` *Two-locus coefficient plot*

Description

Plot two-locus kinship or IBD coefficients as function of the recombination rate.

Usage

```
twoLocusPlot(
  peds,
  coeff = "k11",
  xlab = "Recombination rate",
  ylab = NA,
  col = seq_along(peds),
  lty = 1,
  ...
)
```

Arguments

peds	A list of lists. See details.
coeff	A string identifying which coefficient to compute. See Details for legal values.
xlab, ylab, col, lty	Plotting parameters
...	Further parameters passed on to matplot()

Details

Each entry of peds must be a list with the following (named) entries:

- ped: A ped object
- ids: A pair of labels identifying two members of ped

The coeff parameter must be either a character naming the coefficient to compute, or a function. If a character, it must be one of the following names: "kinship", "phi", "phi11", "k00", "k01", "k02", "k10", "k11", "k12", "k20", "k21" or "k22".

If coeff is a function, it must take three arguments named ped, ids and rho, and produce a single number for each set of input data. See Examples.

The first three are synonymous and indicate the two-locus kinship coefficient. The remaining choices are two-locus IBD coefficients. (See twoLocusIBD().)

Examples

```
##############################
# Classic example of three relationships with equal one-locus coeffs
peds = list(
    GrandParent = list(ped = linearPed(2),    ids = c(1, 5)),
    HalfSib     = list(ped = halfSibPed(),    ids = c(4, 5)),
    Uncle       = list(ped = cousinPed(0, 1), ids = c(3, 6)))

twoLocusPlot(peds, coeff = "kinship")
twoLocusPlot(peds, coeff = "k11")

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

```
peds = list(
    PO = list(ped = nuclearPed(1), ids = c(1,3)),
    S  = list(ped = nuclearPed(2), ids = c(3,4)))

twoLocusPlot(peds, coeff = "kinship")
twoLocusPlot(peds, coeff = "k11")

###############################

ped1 = addChildren(halfSibPed(sex2 = 2), 4, 5, nch = 2)
ped2 = addChildren(addDaughter(nuclearPed(1), 3), 1, 5, nch = 2)
ped3 = addChildren(addDaughter(nuclearPed(2), 4), 3, 6, nch = 2)

peds = list(
   `H-sibs` = list(ped = ped1, ids = leaves(ped1)),
   `G-sibs` = list(ped = ped2, ids = leaves(ped2)),
   `U-sibs` = list(ped = ped3, ids = leaves(ped3))
)
# plotPedList(peds)
twoLocusPlot(peds, coeff = "kinship")

###############################

### Reproducing Fig 2 of Bishop & Williamson (1990)
### This example illustrates `coeff` as a function.

# The coefficient d11(rho) is the conditional probability of IBD = 1
# in the first locus, given IBD = 1 in the second.

G = linearPed(2)
H = halfSibPed()
U = cousinPed(0, removal = 1)
FC = cousinPed(1)
FC1R = cousinPed(1, removal = 1)
SC = cousinPed(2)

peds = list(
    GrandParent = list(ped = G,    ids = c(1, 5)),
    HalfSib     = list(ped = H,    ids = leaves(H)),
    Uncle       = list(ped = U,    ids = leaves(U)),
    FirstCous   = list(ped = FC,   ids = leaves(FC)),
    FirstCous1R = list(ped = FC1R, ids = leaves(FC1R)),
    SecondCous  = list(ped = SC,   ids = leaves(SC)))


d11 = function(ped, ids, rho) {
  twoLocusIBD(ped, ids, rho, coefs = "k11")/kappaIBD(ped, ids)[2]
}

twoLocusPlot(peds, coeff = d11)
```

Index