# Package ‘kinship2’

October 13, 2022

**Version** 1.9.6

**Date** 2022-10-05

**Title** Pedigree Functions

**Depends** R (>= 3.6.0), Matrix, quadprog

**Imports** graphics, stats, methods, knitr

**Description** Routines to handle family data with a pedigree object. The initial purpose was to create correlation structures that describe family relationships such as kinship and identity-by-descent, which can be used to model family data in mixed effects models, such as in the coxme function. Also includes a tool for pedigree drawing which is focused on producing compact layouts without intervention. Recent additions include utilities to trim the pedigree object with various criteria, and kinship for the X chromosome.

**License** GPL (>= 2)

**URL** `https://cran.r-project.org/package=kinship2`

**RoxygenNote** 7.2.1

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Jason Sinnwell [aut, cre],
Terry Therneau [aut],
Daniel Schaid [ctb],
Elizabeth Atkinson [ctb],
Carly Mester [ctb]

**Maintainer** Jason Sinnwell `<sinnwell.jason@mayo.edu>`

**Repository** CRAN

**Date/Publication** 2022-10-05 16:50:02 UTC

## R topics documented:

---

`align.pedigree` *Generate plotting information for a pedigree*

---

## Description

Given a pedigree, this function creates helper matrices that descibe the layout of a plot of the pedigree.

## Usage

```
align.pedigree(ped, packed=TRUE, width=10, align=TRUE, hints=ped$hints)
```

## Arguments

| | |
|---|---|
| `ped` | a pedigree object |
| `packed` | should the pedigree be compressed, i.e., to allow diagonal lines connecting parents to children in order to have a smaller overall width for the plot. |
| `hints` | plotting hints for the pedigree. This is a list with components `order` and `spouse`, the second one is optional. If the hints are missing the `autohint` routine is called to supply an initial guess.<br>The order component is a numeric vector with one element per subject in the pedigree. It determines the relative order of subjects within a sibship, as well |

as the relative order of processing for the founder couples. (For this latter, the female founders are ordered as though they were sisters). The spouse component is a matrix with one row per hinted marriage, usually only a few marriages in a pedigree will need an added hint, for instance reverse the plot order of a husband/wife pair. Each row contains the index of the left spouse, the right hand spouse, and the anchor: 1=left, 2=right, 0=either. Children will preferentially appear under the parents of the anchored spouse.

| | |
|---|---|
| `width` | for a packed output, the minimum width |
| `align` | for a packed pedigree, align children under parents (TRUE), to the extent possible given the page width, or align to to the left margin (FALSE). The latter is mostly used by internal routines. |

## Details

This is an internal routine, used almost exclusively by `plot.pedigree`. The subservient functions `alignped1`, `alignped2`, `alignped3`, and `alignped4` contain the bulk of the computation.

## Value

a structure with components

| | |
|---|---|
| `n` | a vector giving the number of subjects on each horizonal level of the plot |
| `nid` | a matrix with one row for each level, giving the numeric id of each subject plotted. (An value of 17 means the 17th subject in the pedigree). |
| `pos` | a matrix giving the horizontal position of each plot point |
| `fam` | a matrix giving the family id of each plot point. A value of "3" would mean that the two subjects in positions 3 and 4, in the row above, are this subject's parents. |
| `spouse` | a matrix with values 1= subject plotted to the immediate right is a spouse, 2= subject plotted to the immediate right is an inbred spouse, 0 = not a spouse |
| `twins` | optional matrix which will only be present if the pedigree contains twins. It has values 1= sibling to the right is a monozygotic twin, 2= sibling to the right is a dizygotic twin, 3= sibling to the right is a twin of unknown zygosity, 0 = not a twin |

## See Also

`plot.pedigree`, `autohint`

---

`as.data.frame.pedigree` *data.frame from a pedigree object*

---

## Description

Extract the internal data from a pedigree object into a data.frame

### Usage

```
## S3 method for class 'pedigree'
as.data.frame(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | pedigree object |
| `...` | additional arguments passed to internal methods |

### Value

a data.frame with the data necessary to re-create the pedigree, minus special relationships. #' @author Jason Sinnwell

### See Also

`pedigree`

---

## `autohint` *Align a pedigree to print well*

---

### Description

A pedigree structure can contain a `hints` object which helps to reorder the pedigree (e.g. left-to-right order of children within family) so as to plot with minimal distortion. This routine is used to create an initial version of the hints. They can then be modified if desired.

### Usage

```
autohint(ped, hints, packed=TRUE, align=FALSE)
```

### Arguments

| | |
|---|---|
| `ped` | a pedigree structure |
| `hints` | optional hints object. Only the order component is used. |
| `packed` | If TRUE, uniform distance between all individuals at a given level. |
| `align` | these parameters control the extra effort spent trying to align children underneath parents, but without making the pedigree too wide. Set to FALSE to speed up plotting. |

## Details

This routine would not normally be called by a user. It moves children within families, so that marriages are on the "edge" of a set children, closest to the spouse. For pedigrees that have only a single connection between two families this simple-minded approach works surprisingly well. For more complex structures hand-tuning of the hints matrix may be required.

The pedigree in the example below is one where rearranging the founders greatly decreases the number of extra connections. When autohint is called with a a vector of numbers as the second argument, the values for the founder females are used to order the founder families left to right across the plot. The values within a sibship are used as the preliminary order of siblings within a family; this may be changed to move one of them to the edge so as to match up with a spouse. The actual values in the vector are not important, only their order.

## Value

a list containing components `order` and `spouse`

## See Also

pedigree, besthint

## Examples

```
data(testped1)
ped1 <- with(testped1, pedigree(id, father, mother, sex))
plot(ped1, cex=.7, symbolsize=.7)

# rearrange some founders
temp <- 1:nrow(testped1)
temp[76] <- .1
temp[77] <- .2
temp[74] <- .3
temp[60] <- .4
temp[30] <- temp[8] + .1
temp[65] <- temp[4] + .1
temp[14] <- temp[3] + .1
ped1$hints <- autohint(ped1, temp)
plot(ped1, cex=.7)
```

---

`bitSize` *Calculate pedigree bitsize, defined as 2 * # NonFounders - # Founders*

---

## Description

This is a utility function used in pedigree.shrink()

## Usage

```
bitSize(ped)
```

## Arguments

| | |
|---|---|
| `ped` | A pedigree object |

## Value

A list with the following components:

| | |
|---|---|
| `bitSize` | The bitSize of input pedigree |
| `nFounder` | The number of founders in the pedigree |
| `nNonFounder` | The number of nonfounders in the pedgiree |

## See Also

`pedigree.shrink`

---

`familycheck` *Error check for a family classification*

---

## Description

Given a family id vector, also compute the familial grouping from first principles using the parenting data, and compare the results.

## Usage

```
familycheck(famid, id, father.id, mother.id, newfam)
```

## Arguments

| | |
|---|---|
| `famid` | a vector of family identifiers |
| `id` | a vector of unique subject identifiers |
| `father.id` | vector containing the id of the biological father |
| `mother.id` | vector containing the id of the biological mother |
| `newfam` | the result of a call to `makefamid`. If this has allready been computed by the user, adding it as an argument shortens the running time somewhat. |

## Details

The `makefamid` function is used to create a de novo family id from the parentage data, and this is compared to the family id given in the data.

## Value

a data frame with one row for each unique family id in the `famid` argument. Components of the output are

| | |
|---|---|
| `famid` | the family id, as entered into the data set |
| `n` | number of subjects in the family |
| `unrelated` | number of them that appear to be unrelated to anyone else in the entire pedigree set. This is usually marry-ins with no children (in the pedigree), and if so are not a problem. |
| `split` | number of unique "new" family ids. If this is 0, it means that no one in this "family" is related to anyone else (not good); 1 = everythings is fine; 2+= the family appears to be a set of disjoint trees. Are you missing some of the people? |
| `join` | number of other families that had a unique famid, but are actually joined to this one. 0 is the hope. If there are any joins, then an attribute "join" is attached. It will be a matrix with famid as row labels, new-family-id as the columns, and the number of subjects as entries. |

## See Also

`makefamid`, `makekinship`

## Examples

```
# use 2 sample peds
data(sample.ped)
pedAll <- with(sample.ped, pedigree(id, father, mother, sex,
                    affected=cbind(affected, avail), famid=ped))

## check them giving separate ped ids
fcheck.sep <- with(sample.ped, familycheck(ped, id, father, mother))
fcheck.sep

## check assigning them same ped id
fcheck.combined <- with(sample.ped, familycheck(rep(1,nrow(sample.ped)), id, father, mother))
fcheck.combined

#make person 120's father be her son.
sample.ped[20,3] <- 131
fcheck1.bad <- try({with(sample.ped, familycheck(ped, id, father, mother))}, silent=FALSE)

## fcheck1.bad is a try-error
```

---

## `findAvailAffected` *Find a single person to trim from a pedigree whose is available*

---

## Description

Finds one subject from among available non-parents with indicated affection status

## Usage

```
findAvailAffected(ped, avail, affstatus)
```

## Arguments

| | |
|---|---|
| `ped` | A pedigree objects, with id (subject ID), findex (father index), mindex (mother index) |
| `avail` | Vector of availability status (e.g., genotyped) 0/1 or TRUE/FALSE |
| `affstatus` | Vector of affection status 0/1 or TRUE/FALSE. |

## Details

When used within pedigree.shrink, this function is called with the first affected indicator, if the affected item in the pedigree is a matrix of multiple affected indicators.

## Value

A list is returned with the following components

| | |
|---|---|
| `ped` | Dataframe with trimmed subject removed |
| `idTrimmed` | Vector of IDs of trimmed individuals |
| `isTrimmed` | logical value indicating whether pedigree has been trimmed |
| `bitSize` | Bit size of the trimmed pedigree |

## See Also

`pedigree.shrink`

---

## `findAvailNonInform` *Find subjects from a pedigree who are available and uninformative*

---

## Description

Identify subjects to remove from a pedigree who are available but non-informative. This is the second step to remove subjects in pedigree.shrink if the pedigree does not meet the desired bit size.

## Usage

```
findAvailNonInform(ped, avail)
```

## Arguments

| | |
|---|---|
| `ped` | A pedigree object |
| `avail` | Vector of availability status (e.g. genotyped) 0/1 or TRUE/FALSE |

## Value

Vector of subject ids who can be removed by having lowest informativeness.

## See Also

`pedigree.shrink`

---

`findUnavailable` *Find or trim unavailable subjects in a pedigree*

---

## Description

Find the ID of subjects in a pedigree iteratively, as anyone who is not available and does not have an available descendant by successively removing unavailable terminal nodes. pedigree.trim carries out the removal of the subjects identified by findUnavailable.

## Usage

```
findUnavailable(ped, avail)
pedigree.trim(removeID, ped)
```

## Arguments

| | |
|---|---|
| `ped` | A pedigree object with an id, findex, mindex, sex, plus other optional items |
| `avail` | Logical vector of availability status (e.g., genotyped) 0/1 or TRUE/FALSE |
| `removeID` | vector of subject ids of persons to trim from a pedigree |

## Details

Originally written as pedTrim by Steve Iturria, modified by Dan Schaid 2007, and now split into the two separate functions: `findUnavailable()`, and `pedigree.trim()` to do the tasks separately. `findUnavailable()` calls `excludeStrayMarryin` to find stray available marry-ins who are isolated after trimming their unavailable offspring, and excludeUnavailFounders. If the subject ids are character, make sure none of the characters in the ids is a colon (":"), which is a special character used to concatenate and split subjects within the utlity.

## Value

findUnavailable returns a vector of subject ids for who can be removed. pedigree.trim returns a trimmed pedigree object.

## Side Effects

relation matrix from pedigree.trim is trimmed of any special relations that include the subjects to trim.

## See Also

pedigree.shrink,

---

fixParents *Fix details on the parents for children of the pedigree*

---

## Description

Fix the sex of parents, add parents that are missing from the pedigree

## Usage

```
fixParents(id, dadid, momid, sex, missid = 0)
```

## Arguments

| | |
|---|---|
| `id` | Identification variable for individual |
| `dadid` | Identification variable for father. Founders parents should be coded to NA, or another value specified by missid. |
| `momid` | Identification variable for mother. Founders parents should be coded to NA, or another value specified by missid. |
| `sex` | Gender of individual noted in 'id'. Either character ("male","female","unknown","terminated") or numeric (1="male", 2="female", 3="unknown", 4="terminated") data is allowed. For character data the string may be truncated, and of arbitrary case. |
| `missid` | The founders are those with no father or mother in the pedigree. The `dadid` and `momid` values for these subjects will either be NA or the value of this variable. The default for `missid` is 0 if the `id` variable is numeric, and "" (the empty string) otherwise. |

## Details

First look to add parents whose ids are given in momid/dadid. Second, fix sex of parents. Last look to add second parent for children for whom only one parent id is given.

## Value

A data.frame with id, dadid, momid, sex as columns

## Author(s)

Jason Sinnwell

## See Also

pedigree

## Examples

```
test1char <- data.frame(id=paste("fam", 101:111, sep=""),
  sex=c("male","female")[c(1,2,1,2,1, 1,2, 2,1,2, 1)],
  father=c(0,0,"fam101","fam101","fam101", 0,0,"fam106","fam106","fam106", "fam109"),
  mother=c(0,0,"fam102","fam102","fam102", 0,0,"fam107","fam107","fam107", "fam112"))
test1newmom <- with(test1char, fixParents(id, father, mother, sex, missid="0"))
newped <- with(test1newmom, pedigree(id, dadid, momid, sex, missid="0"))
as.data.frame(newped)
```

---

`ibdMatrix` *Create an IBD matrix*

---

## Description

Transform identity by descent (IBD) matrix data from the form produced by external programs such as SOLAR into the compact form used by the coxme and lmekin routines.

## Usage

```
ibdMatrix(id1, id2, x, idmap, diagonal)
```

## Arguments

| | |
|---|---|
| `id1, id2` | pairs of subject identifiers |
| `x` | the IBD value for that pair |
| `idmap` | an optional 2 column matrix or data frame whose first element is the internal value (as found in `id1` and `id2`), and whose second element will be used for the dimnames of the result |
| `diagonal` | optional value for the diagonal element. If present, any missing diagonal elements in the input data will be set to this value. |

## Details

The IBD matrix for a set of n subjects will be an n by n symmetric matrix whose i,j element is the contains, for some given genetic location, a 0/1 indicator of whether 0, 1/2 or 2/2 of the alleles for i and j are identical by descent. Fractional values occur if the IBD fraction must be imputed. The diagonal will be 1. Since a large fraction of the values will be zero, programs such as Solar return a data set containing only the non-zero elements. As well, Solar will have renumbered the subjects as 1:n in such a way that families are grouped together in the matrix; a separate index file contains the mapping between this new id and the original one. The final matrix should be labeled with the original identifiers.

## Value

a sparse matrix of class `dsCMatrix`. This is the same form used for kinship matrices.

## See Also

`kinship`, `Matrix`

---

`kindepth` *Compute the depth of each subject in a pedigree*

---

## Description

Mark each person as to their depth in a pedigree; 0 for a founder, otherwise depth = 1 + max(father's depth, mother's depth)

## Usage

```
kindepth(id, dad.id, mom.id, align = FALSE)
```

## Arguments

| | |
|---|---|
| `id` | Identification code for each individual |
| `dad.id` | Id code for the father |
| `mom.id` | Id code for the mother |
| `align` | If align=T, go one step further and try to make both parents of each child have the same depth. (This is not always possible). It helps the drawing program by lining up pedigrees that "join in the middle" via a marriage. |

## Details

In the case of an inbred pedigree a perfect alignment obeying `extra=TRUE` may not exist.

## Value

an integer vector containing the depth for each subject

## Author(s)

Terry Therneau

## See Also

`plot.pedigree`

## kinship — *Compute a kinship matrix*

### Description

Compute the kinship matrix for a set of related autosomal subjects. The function is generic, and can accept a pedigree, pedigreeList, or vector as the first argument.

### Usage

```
kinship(id, ...)
## S3 method for class 'pedigree'
kinship(id, chrtype="autosome", ...)
## S3 method for class 'pedigreeList'
kinship(id, chrtype="autosome", ...)
## Default S3 method:
kinship(id, dadid, momid, sex, chrtype="autosome", ...)
```

### Arguments

| | |
|---|---|
| `id` | either a pedigree object, pedigreeList object, or a vector of subject identifiers. Subject identifiers may be numeric or character. |
| `dadid` | for each subject, the identifier of the biological father. This is only used if `id` is a vector. |
| `momid` | for each subject, the identifier of the biological mother. This is only used if `id` is a vector. |
| `sex` | vector of sex values coded as 1=male, 2=female |
| `chrtype` | chromosome type. The currently supported types are "autosome" and "X" or "x". |
| `...` | Any number of optional arguments |

### Details

The function will usually be called with a pedigree or pedigreeList; the third form is provided for backwards compatability with an earlier release of the library that was less capable. The first argument is named `id` for the same reason. Note that when using the third form any information on twins is not available to the function.

When called with a pedigreeList, i.e., with multiple families, the routine will create a block-diagonal-symmetric sparse matrix object of class `dsCMatrix`. Since the [i,j] value of the result is 0 for any two unrelated individuals i and j and a `Matrix` utilizes sparse representation, the resulting object is often orders of magnitude smaller than an ordinary matrix. When `kinship` is called with a single pedigree an ordinary matrix is returned.

Two genes G1 and G2 are identical by descent (IBD) if they are both physical copies of the same ancestral gene; two genes are identical by state if they represent the same allele. So the brown eye gene that I inherited from my mother is IBD with hers; the same gene in an unrelated individual is not.

The kinship coefficient between two subjects is the probability that a randomly selected allele from a locus will be IBD between them. It is obviously 0 between unrelated individuals. For an autosomal site and no inbreeding it will be 0.5 for an individual with themselves, .25 between mother and child, .125 between an uncle and neice, etc.

The computation is based on a recursive algorithm described in Lange, which assumes that the founder alleles are all independent.

## Value

a matrix of kinship coefficients.

## References

K Lange, Mathematical and Statistical Methods for Genetic Analysis, Springer-Verlag, New York, 1997.

## See Also

`pedigree`, `makekinship`,`makefamid`

## Examples

```
test1 <- data.frame(id  =c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14),
                    mom =c(0, 0, 0, 0, 2, 2, 4, 4, 6,  2,  0,  0, 12, 13),
                    dad =c(0, 0, 0, 0, 1, 1, 3, 3, 3,  7,  0,  0, 11, 10),
                    sex =c(0, 1, 0, 1, 0, 1, 0, 1, 0,  0,  0,  1,  1,  1))
tped <- with(test1, pedigree(id, dad, mom, sex))
round(8*kinship(tped))
```

---

`kinship2` *The kinship2 package for pedigree data*

---

## Description

The kinship2 package for pedigree data

## Details

The package download, NEWS, and README are available on CRAN: `https://cran.r-project.org/package=kinship2`

## Functions

Below are listed some of the most widely used functions available in `arsenal`:

`pedigree`: Contstructor of the pedigree class, given identifiers, sex, affection status(es), and special relationships

`kinship`: Calculates the kinship matrix, the probability having an allele sampled from two individuals be the same via IBD.

`plot.pedigree`: Plot method for a pedigree object. Allows extra information to be included in the id under the plot symbol

`legendPlot`: Special version of plot.pedigree, with an informative legend along the bottom of the figure

`pedigree.shrink`: Shrink a pedigree to a specific bit size, removing non-informative members first.

`bitSize`: Approximate the output from SAS's `PROC FREQ` procedure when using the `/list` option of the `TABLE` statement.

## Data

`sample.ped`: Pedigree example data sets with two pedigrees

`minnbreast`: Larger cohort of pedigrees from MN breast cancer study

## Examples

```
library(kinship2)
```

---

`legendPlot` *Legend Pedigree Plot*

---

## Description

Pedigree plot with ready-made legend along the bottom of the page to represent colors and affection statuses

## Usage

```
legendPlot(
  x,
  id = x$id,
  affected = x$affected,
  affected.label = NULL,
  col = 1,
  col.label = NULL,
  symbolsize = 0.75,
  cex = 0.5,
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | Pedigree data frame with ped (pedigree id), id (id of individual), father (id of father), mother (id of mother), sex, affected (affection status), and avail (DNA availability). |
| `id` | Optional, a character string to replace the correspinding id for persons in the pedigree |
| `affected` | A variable indicating affection status. A multi-column matrix can be used to give the status with respect to multiple traits. Logical, factor, and integer types are converted to 0/1 representing unaffected and affected, respectively. NAs are considered missing. |
| `affected.label` | Set labels for affection statuses |
| `col` | Colors for the plot symbol for each individual |
| `col.label` | Named vector, with elements matching the unique color codes, the names are the labels used in the legend. |
| `symbolsize` | Size of symbols (circle/square/triangle). Default is 1.0 |
| `cex` | Character expansion size for labels and ids. Default is 1.0 |
| `...` | Character expansion size for labels and ids. Default is 1.0 |

## Author(s)

Jason Sinnwell, code contributed by Sara Achenbach

## See Also

`pedigree`, `plot.pedigree`

## Examples

```
## Not run:
data(sample.ped)
pedAll <- pedigree(sample.ped$id, sample.ped$father,
   sample.ped$mother, sample.ped$sex,
   affected=cbind(sample.ped$affected, sample.ped$avail),
   famid=sample.ped$ped)
ped1 <- pedAll["1"]
legendPlot(ped1,  affected.label=c("cancer","available"))

## End(Not run)
```

| makefamid | *Construct a family id from pedigree information* |
|---|---|

## Description

Create a vector of length n, giving the family "tree" number of each subject. If the pedigree is totally connected, then everyone will end up in tree 1, otherwise the tree numbers represent the disconnected subfamilies. Singleton subjects give a zero for family number.

## Usage

```
makefamid(id, father.id, mother.id)
```

## Arguments

| | |
|---|---|
| `id` | Identifier for each subject in the set of pedigrees |
| `father.id` | Identifier for the father. This will be 0 or "" for a founder. |
| `mother.id` | Identifer for the mother. |

## Details

This command is depricated. The kinship command now can be applied directly to pedigreeList objects.

## Value

An integer vector giving family groupings

## Author(s)

Terry Therneau

## See Also

`makekinship`

## makekinship *Create a sparse kinship matrix*

### Description

Compute the overall kinship matrix for a collection of families, and store it efficiently.

### Usage

```
makekinship(famid, id, father.id, mother.id, unrelated=0)
```

### Arguments

| | |
|---|---|
| famid | a vector of family identifiers |
| id | a vector of unique subject identifiers |
| father.id | for each subject, the identifier of their biolgical father |
| mother.id | for each subject, the identifier of thier biological mother |
| unrelated | subjects with this family id are considered to be unrelated singletons, i.e., not related to each other or to anyone else. |

### Details

This command is depricated. The kinship command now can be applied directly to pedigreeList objects.

### Value

a sparse kinship matrix of class bdsmatrix

### See Also

kinship, makefamid

### Examples

```
# Data set from a large family study of breast cancer
#  there are 26050 subjects in the file, from 426 families
## Not run:
> table(cdata$sex)
     F     M
 12699 13351
> length(unique(cdata$famid))
[1] 426

> kin1 <- makekinship(cdata$famid, cdata$gid, cdata$dadid, cdata$momid)
> dim(kin1)
[1] 26050 26050
> class(kin1)
```

```
[1] "bdsmatrix"
# The next line shows that few of the elements of the full matrix are >0
> length(kin1@blocks)/ prod(dim(kin1))
[1] 0.00164925

# kinship matrix for the females only
> femid <- cdata$gid[cdata$sex=='F']
> femindex <- !is.na(match(dimnames(kin1)[[1]], femid))
> kin2 <- kin1[femindex, femindex]
#
# Note that "femindex <- match(femid, dimnames(kin1)[[1]])" is wrong, since
#  then kin1[femindex, femindex] might improperly reorder the rows/cols
#  (if families were not contiguous in cdata).
# However sort(match(femid, dimnames(kin1)[[1]])) would be okay.

## End(Not run)
```

## `minnbreast` *Minnesota Breast Cancer Study*

### Description

Data from the Minnesota Breast Cancer Family Study. This contains extended pedigrees from 426 families, each identified by a single proband in 1945-52, with follow up for incident breast cancer.

### Usage

`data(minnbreast)`

### Format

minnbreast: A data frame with 28081 observations, one line per subject, on the following 14 variables.

`id` subject identifier

`proband` if 1, this subject is one of the original 426 probands

`fatherid` identifier of the father, if the father is part of the data set; zero otherwise

`motherid` identifier of the mother, if the mother is part of the data set; zero otherwise

`famid` family identifier

`endage` age at last follow-up or incident cancer

`cancer` 1= breast cancer (females) or prostate cancer (males), 0=censored

`yob` year of birth

`education` amount of education: 1-8 years, 9-12 years, high school graduate, vocational education beyond high school, some college but did not graduate, college graduate, post-graduate education, refused to answer on the questionairre `marstat` marital status: married, living with someone in a marriage-like relationship, separated or divorced, widowed, never married, refused to answer the questionaiire

`everpreg` ever pregnant: never pregnant at the time of baseline survey, ever pregnant at the time of baseline survey

`parity` number of births

`nbreast` number of breast biopsies

**sex** M or F

**bcpc** part of one of the families in the breast/prostate cancer substudy: 0=no, 1=yes. Note that subjects who were recruited to the overall study after the date of the BP substudy are coded as zero.

## Details

The original study was conducted by Dr. Elving Anderson at the Dight Institute for Human Genetics at the University of Minnesota. From 1944 to 1952, 544 sequential breast cancer cases seen at the University Hospital were enrolled, and information gathered on parents, siblings, offspring, aunts/uncles, and grandparents with the goal of understanding possible familial aspects of brest cancer. In 1991 the study was resurrected by Dr. Tom Sellers. Of the original 544 he excluded 58 prevalent cases, along with another 19 who had less than 2 living relatives at the time of Dr Anderson's survey. Of the remaining 462 families 10 had no living members, 23 could not be located and 8 refused, leaving 426 families on whom updated pedigrees were obtained. This gave a study with 13351 males and 12699 females (5183 marry-ins). Primary questions were the relationship of early life exposures, breast density, and pharmacogenomics on incident breast cancer risk.

For a subset of the families data was gathered on prostate cancer risk for male subjects via questionairres sent to men over 40. Other than this, data items other than parentage are limited to the female subjects.

In ___ a second phase of the study was instituted. The pedigrees were further extended to the numbers found in this data set, and further data gathered by questionairre.

## Source

Authors of the study

## References

Epidemiologic and genetic follow-up study of 544 Minnesota breast cancer families: design and methods. Sellers TA, Anderson VE, Potter JD, Bartow SA, Chen PL, Everson L, King RA, Kuni CC, Kushi LH, McGovern PG, et al. Genetic Epidemiology, 1995; 12(4):417-29.

Evaluation of familial clustering of breast and prostate cancer in the Minnesota Breast Cancer Family Study. Grabrick DM, Cerhan JR, Vierkant RA, Therneau TM, Cheville JC, Tindall DJ, Sellers TA. Cancer Detect Prev. 2003; 27(1):30-6.

Risk of breast cancer with oral contraceptive use in women with a family history of breast cancer. Grabrick DM, Hartmann LC, Cerhan JR, Vierkant RA, Therneau TM, Vachon CM, Olson JE, Couch FJ, Anderson KE, Pankratz VS, Sellers TA. JAMA. 2000; 284(14):1791-8.

## Examples

```
data(minnbreast)
breastped <- with(minnbreast, pedigree(id, fatherid, motherid, sex,
                  status=(cancer& !is.na(cancer)), affected=proband,
                   famid=famid))
print(breastped["8"])
print(breastped[8])
#plot(breastped["8"])  #plot family 8, proband is solid, slash for cancers
#Note that breastped[8] is a different family, since ids are not 1,2,3,...
```

---

`pedigree` *Create a pedigree or pedigreeList object*

---

## Description

Create a pedigree or pedigreeList object

## Usage

```
pedigree(id, dadid, momid, sex, affected, status, relation, famid, missid)

## S3 method for class 'pedigreeList'
x[..., drop = FALSE]

## S3 method for class 'pedigree'
x[..., drop = FALSE]

## S3 method for class 'pedigree'
print(x, ...)

## S3 method for class 'pedigreeList'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `id` | Identification variable for individual |
| `dadid` | Identification variable for father. Founders' parents should be coded to NA, or another value specified by missid. |
| `momid` | Identification variable for mother. Founders' parents should be coded to NA, or another value specified by missid. |
| `sex` | Gender of individual noted in 'id'. Either character ("male","female", "unknown","terminated") or numeric (1="male", 2="female", 3="unknown", 4="terminated") data is allowed. For character data the string may be truncated, and of arbitrary case. |

| | |
|---|---|
| `affected` | A variable indicating affection status. A multi-column matrix can be used to give the status with respect to multiple traits. Logical, factor, and integer types are converted to 0/1 representing unaffected and affected, respectively. NAs are considered missing. |
| `status` | Censor/Vital status (0="censored", 1="dead") |
| `relation` | A matrix with 3 required columns (id1, id2, code) specifying special relationship between pairs of individuals. Codes: 1=Monozygotic twin, 2=Dizygotic twin, 3=Twin of unknown zygosity, 4=Spouse. (The last is necessary in order to place a marriage with no children into the plot.) If famid is given in the call to create pedigrees, then famid needs to be in the last column of the relation matrix. Note for tuples of >= 3 with a mixture of zygosities, the plotting is limited to showing pairwise zygosity of adjacent subjects, so it is only necessary to specify the pairwise zygosity, in the order the subjects are given or appear on the plot. |
| `famid` | An optional vector of family identifiers. If it is present the result will contain individual pedigrees for each family in the set, which can be extacted using subscripts. Individuals need to have a unique id *within* family. |
| `missid` | The founders are those with no father or mother in the pedigree. The dadid and momid values for these subjects will either be NA or the value of this variable. The default for missid is 0 if the id variable is numeric, and "" (empty string) otherwise. |
| `x` | pedigree object in print and subset methods |
| `...` | optional arguments passed to internal functions |
| `drop` | logical, used in subset function for dropping dimensionanlity |

## Value

An object of class `pedigree` or `pedigreeList` Containing the following items: famid id findex mindex sex affected status relation @examples data(minnbreast) bpeds <- with(minnbreast, pedigree(id, fatherid, motherid, sex, affected=proband, famid=famid)) bped.id8 <- bpeds['8'] print(bped.id8) ## show this pedigree with mixed zygosity quadruplets rel8 <- data.frame(id1=c(137,138,139), id2=c(138,139,140), code=c(1,2,2)) bped.id8 <- with(minnbreast[minnbreast$famid==8,], pedigree(id, fatherid, motherid, sex, affected=proband, relation=rel8)) print(bped.id8)

## Author(s)

Terry Therneau

## See Also

`kinship`, `plot.pedigree`, `autohint`

## pedigree.legend *plot a legend for a pedigree*

### Description

circular legend for a pedigree as a key to the affection statuses

### Usage

```
pedigree.legend(
  ped,
  labels = dimnames(ped$affected)[[2]],
  edges = 200,
  radius = NULL,
  location = "bottomright",
  new = TRUE,
  density = c(-1, 35, 65, 20),
  angle = c(90, 65, 40, 0),
  ...
)
```

### Arguments

| | |
|---|---|
| `ped` | Pedigree data frame with ped (pedigree id), id (id of individual), father (id of father), mother (id of mother), sex, affected (affection status), and avail (DNA availability). |
| `labels` | names for the affected indicators |
| `edges` | Number of edges for each polygon. Higher numbers give better resolution for the circle |
| `radius` | radius (inches) of the circle |
| `location` | similar to how the location of a base-R legend is given, used only if new=TRUE. A character string indicating which of the four corners to plot the legend, given by "bottomright", "bottomleft", "topleft", or "topright". |
| `new` | Logical. If TRUE, plot the legend on the current plot. Otherwise, plot on a separate plot. |
| `density` | Density of lines shaded in sections of the circle. These match the density settings for the plot.pedigree function. |
| `angle` | The angle at which lines are shaded in sections of the circle. These match the angles for the plot.pedigree function. |
| `...` | optional parameters for the plot function that apply to text |

### Author(s)

Jason Sinnwell

## See Also

`pedigree`, `plot.pedigree`

## Examples

```
## Not run:
data(sample.ped)
fam1 <- sample.ped[sample.ped$ped==1,]
ped1 <- with(fam1, pedigree(id, father, mother, sex,
             affected=cbind(avail,affected)))
plot(ped1)
pedigree.legend(ped1, location="bottomright", radius=.8)
pedigree.legend(ped1, location="topleft", radius=.6, cex=1.2)
pedigree.legend(ped1, new=FALSE)

## End(Not run)
```

---

`pedigree.shrink` *Shrink pedigree object*

---

## Description

Shrink pedigree object to specified bit size with priority placed on trimming uninformative subjects. The algorithm is useful for getting a pedigree condensed to a minimally informative size for algorithms or testing that are limited by size of the pedigree.

## Usage

```
pedigree.shrink(ped, avail, affected = NULL, maxBits = 16)

## S3 method for class 'pedigree.shrink'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `ped` | Pedigree object created by the pedigree function, |
| `avail` | vector of binary availability status (0/1), i.e. having data, or sample available |
| `affected` | vector of binary affected status (0/1/NA). If NULL, uses first column of the pedigree object affected matrix. |
| `maxBits` | Optional, the bit size for which to shrink the pedigree |
| `x` | pedigree.shrink object used in method functions |
| `...` | optional arguments passed to internal functions |

### Details

Iteratively remove subjects from the pedigree. The random removal of members was previously controlled by a seed argument, but we remove this, forcing users to control randomness outside the function. First remove uninformative subjects, i.e., unavailable (not genotyped) with no available descendants. Next, available terminal subjects with unknown phenotype if both parents available. Last, iteratively shrinks pedigrees by preferentially removing individuals (chosen at random if there are multiple of the same status): 1. Subjects with unknown affected status, 2. Subjects with unaffected affected status 3. Affected subjects.

### Author(s)

Original by Dan Schaid, updated to kinship2 by Jason Sinnwell

### See Also

`pedigree`, `plot.pedigree.shrink`

### Examples

```
data(sample.ped)
pedAll <- pedigree(sample.ped$id, sample.ped$father, sample.ped$mother,
  sample.ped$sex, affected=cbind(sample.ped$affected, sample.ped$avail),
  famid=sample.ped$ped)
ped1 <- pedAll['1']
pedigree.shrink(ped1, maxBits=12, avail=ped1$affected[,2])
```

---

## `pedigree.unrelated` *Determine set of maximum number of unrelated available subjects from a pedigree*

---

### Description

Determine set of maximum number of unrelated available subjects from a pedigree, given vectors id, father, and mother for a pedigree structure, and status vector of T/F for whether each subject is available (e.g. has DNA)

### Usage

```
pedigree.unrelated(ped, avail)
```

### Arguments

| | |
|---|---|
| `ped` | A pedigree objects with unique id, father index, mother index |
| `avail` | Vector of availability status (e.g., genotyped) 0/1 or TRUE/FALSE |

## Details

This is a greedy algorithm that uses the kinship matrix, sequentially removing rows/cols that are non-zero for subjects that have the most number of zero kinship coefficients (greedy by choosing a row of kinship matrix that has the most number of zeros, and then remove any cols and their corresponding rows that are non-zero. To account for ties of the count of zeros for rows, a random choice is made. Hence, running this function multiple times can return different sets of unrelated subjects.

## Value

A vector of the ids of subjects that are unrelated.

## See Also

kinship, pedigree

## Examples

```
data(sample.ped)
fam1 <- sample.ped[sample.ped$ped==1,]


ped1 <- pedigree(fam1$id, fam1$father, fam1$mother,
                 fam1$sex, fam1$affected, fam1$avail)

## to see plot:
## plot.pedigree(ped1, align=FALSE)
id1 <- pedigree.unrelated(ped1, avail=fam1$avail)

id1
## some possible vectors
##[1] "110" "113" "133" "109"
##[1] "113" "118" "141" "109"
##[1] "113" "118" "140" "109"
##[1] "110" "113" "116" "109"
##[1] "113" "133" "141" "109"


fam2 <- sample.ped[sample.ped$ped==2,]

ped2 <- pedigree(fam2$id, fam2$father, fam2$mother,
               fam2$sex, fam2$affected, fam2$avail)

## to see plot:
## plot.pedigree(ped2, align=FALSE)

id2 <- pedigree.unrelated(ped2, avail=fam2$avail)

## some possible vectors
##[1] "203" "207"
##[1] "203" "204"
```

```
##[1] "201" "203"
##[1] "214" "203"
id2
```

---

plot.pedigree *plot pedigrees*

---

## Description

plot objects created with the pedigree function

## Usage

```
## S3 method for class 'pedigree'
plot(x, id = x$id, status = x$status,
                         affected = x$affected,
                         cex = 1, col = 1,
                         symbolsize = 1, branch = 0.6,
                         packed = TRUE, align = c(1.5,2), width = 8,
                         density=c(-1, 35,65,20), #mar=c(3.1, 1, 3.1, 1),
                         angle=c(90,65,40,0), keep.par=FALSE,
                         subregion, pconnect=.5, ...)
```

## Arguments

| | |
|---|---|
| `x` | object created by the function pedigree. |
| `id` | id variable - used for labeling. |
| `status` | can be missing. If it exists, 0=alive/missing and 1=death. |
| `affected` | vector, or matrix with up to 4 columns for affected indicators. Subject's symbol is divided into sections for each status, shaded if indicator is 1, not-shaded for 0, and symbol "?" if missing (NA) |
| `cex` | controls text size. Default=1. |
| `col` | color for each id. Default assigns the same color to everyone. |
| `symbolsize` | controls symbolsize. Default=1. |
| `branch` | defines how much angle is used to connect various levels of nuclear families. |
| `packed` | default=T. If T, uniform distance between all individuals at a given level. |
| `align` | these parameters control the extra effort spent trying to align children underneath parents, but without making the pedigree too wide. Set to F to speed up plotting. |
| `width` | default=8. For a packed pedigree, the minimum width allowed in the realignment of pedigrees. |
| `density` | defines density used in the symbols. Takes up to 4 different values. |
| `mar` | margin parmeters, as in the `par` function |

| | |
|---|---|
| `angle` | defines angle used in the symbols. Takes up to 4 different values. |
| `keep.par` | Default = F, allows user to keep the parameter settings the same as they were for plotting (useful for adding extras to the plot) |
| `subregion` | 4-element vector for (min x, max x, min depth, max depth), used to edit away portions of the plot coordinates returned by align.pedigree |
| `pconnect` | when connecting parent to children the program will try to make the connecting line as close to vertical as possible, subject to it lying inside the endpoints of the line that connects the children by at least `pconnect` people. Setting this option to a large number will force the line to connect at the midpoint of the children. |
| `...` | Extra options that feed into the plot function. |

## Details

Two important parameters control the looks of the result. One is the user specified maximum width. The smallest possible width is the maximum number of subjects on a line, if the user's suggestion is too low it is increased to 1+ that amount (to give just a little wiggle room). To make a pedigree where all children are centered under parents simply make the width large enough, however, the symbols may get very small.

The second is `align`, a vector of 2 alignment parameters $a$ and $b$. For each set of siblings at a set of locations `x` and with parents at `p=c(p1,p2)` the alignment penalty is

$$(1/k^a)\sum i = 1k[(x_i - (p1 + p2)/2)]^2$$

sum(x- mean(p))^2/(k^a) where k is the number of siblings in the set. when $a=1$ moving a sibship with $k$ sibs one unit to the left or right of optimal will incur the same cost as moving one with only 1 or two sibs out of place. If $a=0$ then large sibships are harder to move than small ones, with the default value $a=1.5$ they are slightly easier to move than small ones. The rationale for the default is as long as the parents are somewhere between the first and last siblings the result looks fairly good, so we are more flexible with the spacing of a large family. By tethering all the sibs to a single spot they are kept close to each other. The alignment penalty for spouses is $b(x_1 - x_2)^2$, which tends to keep them together. The size of $b$ controls the relative importance of sib-parent and spouse-spouse closeness.

## Value

an invisible list containing

| | |
|---|---|
| `plist` | a list that contains all the position information for plotting the pedigree. This will useful for further functions (yet unwritten) for manipulating the plot, but likely not to an ordinary user. |
| `x,y` | the x an and y plot coordinates of each subject in the plot. The coordinate is for the top of the plotted symbol. These will be in the same order as the input pedigree. If someone in the pedigree does not appear in the plot their coordinates will be NA. If they appear multiple times one of the instances is chosen. (Which one is a function of the order in which the pedigree was constructed.) |
| `boxh` | the height of the symbol, in user coordinates |
| `boxw` | the width of the symbol |
| `call` | a copy of the call that generated the plot |

### Side Effects

creates plot on current plotting device.

### See Also

pedigree

### Examples

```
data(sample.ped)

pedAll <- pedigree(sample.ped$id, sample.ped$father, sample.ped$mother,
       sample.ped$sex,  #affected=sample.ped$affected,
       affected=cbind(sample.ped$affected, sample.ped$avail),
       famid=sample.ped$ped)

ped2 <- pedAll['2']

print(ped2)


## plot(ped2)
```

---

## `plot.pedigree.shrink` *plot pedigree.shrink object that is a shrunk pedigree object*

---

### Description

Plot the pedigree object that is the trimmed pedigree.shrink object along with colors based on availability and affection status.

### Usage

```
## S3 method for class 'pedigree.shrink'
plot(x, bigped=FALSE, title="", xlegend="topright", ...)
```

### Arguments

| | |
|---|---|
| `x` | A pedigree.shrink object, which contains a pedigree object and information about which subject was removed. |
| `bigped` | Logical value indicating whether pedigree should be compacted to fit in plotting region. If T, then packed=F is used in pedigree() along with smaller symbol sizes. |
| `title` | Optional plot title |
| `xlegend` | The x argument for the legend command, which allows coordinates or, more conveniently, options such as "topright", "right", "left", "bottomleft", etc., which is useful for pedigrees that cover most of the plot region. |
| `...` | Optional arguments to plot method |

## See Also

pedigree.shrink,

## Examples

```
data(sample.ped)

fam2 <- sample.ped[sample.ped$ped==2,]
ped2 <- pedigree(fam2$id, fam2$father, fam2$mother, fam2$sex,
                 fam2$affected, fam2$avail)

shrink2 <- pedigree.shrink(ped2,avail=fam2$avail)
shrink2

plot(ped2)

plot.pedigree.shrink(shrink2, title="Sample Pedigree 2")
```

---

`sample.ped` *Two example pedigrees*

---

## Description

Two pedigrees of different size for testing pedigree operations

## Usage

```
data(sample.ped)
```

## Format

A data frame with 55 observations on the following 7 variables.

| | |
|---|---|
| `ped` | pedigree id |
| `id` | subject id, unique to each ped |
| `father` | id of the subject's father |
| `mother` | id of the subject's mother |
| `sex` | 1=male, 2=female, 3=unknown |
| `affected` | affection status; 0=unaffected, 1=affected, NA=unknown |
| `avail` | is DNA data available; 0=unavailable, 1=available |

## Examples

```
data(sample.ped)
```

## `testped1` *testped1*

### Description

Sample pedigree

### Usage

```
data(testped1)
```

### Format

A data frame with 79 subjects in one family with the following variables.

`id` subject id, unique to each ped

`father` id of the subject's father

`mother` id of the subject's mother

`sex` 1=male, 2=female

### Examples

```
data(testped1)
testped1[1:20,]
```

# Index