Package: clipp (via r-universe)

November 22, 2026

Type Package

Title Calculating Likelihoods by Pedigree Paring

Version 2.0.0

Description A fast and general implementation of the Elston-Stewart algorithm that can calculate the likelihoods of large and complex pedigrees. References for the Elston-Stewart algorithm are Elston & Stewart (1971) <doi:10.1159/000152448>, Lange & Elston (1975) <doi:10.1159/000152714> and Cannings et al. (1978) <doi:10.2307/1426718>. This package is supported by the Genetic Epidemiology Research Alliance, a Centre of Research Excellence funded by Australia's National Health and Medical Research Council.

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

Suggests knitr, rmarkdown

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

Imports graphics (>= 3.5.0), grDevices (>= 3.5.0), methods (>= 3.5.0), stats4 (>= 3.5.0)

NeedsCompilation no

Author James Dowty [aut, cre], Kevin Wong [aut], Genetic Epidemiology Research Alliance [fnd] (https://www.gera.org.au/, A Centre of Research Excellence funded by Australia's National Health and Medical Research Council)

Maintainer James Dowty <jgdowty@gmail.com>

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

Date/Publication 2026-11-22 08:40:02 UTC

RemoteUrl https://github.com/cran/clipp

RemoteRef HEAD

RemoteSha 30949797a41b6424d449c475d98619e284963990

Contents

clipp-package . . . 2
combine_loci . . . 3
dat_large . . . 5
dat_seg . . . 6
dat_small . . . 7
fit_seg . . . 7
geno_freq_monogenic . . . 8
geno_freq_phased . . . 9
geno_freq_polygenic . . . 10
genotype_probabilities . . . 12
inc_seg . . . 14
pedigree_loglikelihood . . . 15
penet_large . . . 18
penet_small . . . 19
plot_pedigree . . . 19
plot_pedigree_adv . . . 22
segregation_analysis . . . 25
trans_monogenic . . . 30
trans_phased . . . 31
trans_polygenic . . . 33

Index **35**

`clipp-package` *clipp: Calculate Likelihoods by Pedigree Paring*

Description

`clipp` provides a fast and general implementation of the Elston-Stewart algorithm, and can calculate the log-likelihoods of large and complex pedigrees, including those with loops. General references for the Elston-Stewart algorithm are (Elston & Stewart, 1971), (Lange & Elston, 1975) and (Cannings et al., 1978). This package is supported by the Genetic Epidemiology Research Alliance, https://www.gera.org.au/, a Centre of Research Excellence funded by Australia's National Health and Medical Research Council.

Details

The main function is `pedigree_loglikelihood`, which calculates the pedigree likelihood on page 117 of (Lange, 2002) for almost any choice of genotype frequencies, transmission matrix and penetrance matrix. Helper functions are provided to calculate the genotype frequencies and transmission matrices for genetic models that often arise in applications. The function `genotype_probabilities` calculates genotype probabilities for a target person within a family, given the family's phenotypes.

The `clipp` package can handle pedigree loops, such as those caused by inbreeding or by two sisters having children with two brothers from an unrelated family (see (Totir et al., 2009) for a precise definition). However, pedigrees with more than a few loops could greatly reduce the speed of the calculation.

It is feasible to apply clipp to very large families. For instance, in the examples for pedigree_loglikelihood, the log-likelihood of one family with approximately 10,000 members is calculated in less than one minute on a standard desktop computer. Numerical issues will eventually limit the family size, though clipp takes care to avoid arithmetic underflow and other issues.

Author(s)

Maintainer: James Dowty <jgdowty@gmail.com>

Authors:

- James Dowty <jgdowty@gmail.com>
- Kevin Wong <wongck.kevin@gmail.com>

Other contributors:

- Genetic Epidemiology Research Alliance (https://www.gera.org.au/, A Centre of Research Excellence funded by Australia's National Health and Medical Research Council) [funder]

References

Cannings C, Thompson E, Skolnick M. Probability functions on complex pedigrees. Advances in Applied Probability, 1978;10(1):26-61.

Elston RC, Stewart J. A general model for the genetic analysis of pedigree data. Hum Hered. 1971;21(6):523-542.

Lange K. Mathematical and Statistical Methods for Genetic Analysis (second edition). Springer, New York. 2002.

Lange K, Elston RC. Extensions to pedigree analysis I. Likehood calculations for simple and complex pedigrees. Hum Hered. 1975;25(2):95-105.

Totir LR, Fernando RL, Abraham J. An efficient algorithm to compute marginal posterior genotype probabilities for every member of a pedigree with loops. Genet Sel Evol. 2009;41(1):52.

---

combine_loci *Combine the genotype frequencies and transition matrices of two genetic loci*

Description

A function to calculate the genotype frequencies and the transition matrices for the joint genotypes of two unlinked genetic loci in linkage equilibrium, given the corresponding objects for the separate loci. The results from this function can be used as inputs to pedigree_loglikelihood or genotype_probabilities to model the combined effect of the two loci on phenotypes.

Usage

```
combine_loci(geno_freq1, geno_freq2, trans1, trans2, annotate = FALSE)
```

Arguments

geno_freq1	A vector of strictly positive numbers that sum to 1, with geno_freq1[i] interpreted as the population genotype frequency of the ith possible genotype at a genetic locus (locus 1). When annotate is TRUE, the names of the genotypes at locus 1 will be taken to be names(geno_freq1) or, if names(geno_freq1) is NULL, to be 1:length(geno_freq1).
geno_freq2	Similar to geno_freq1 (above) but interpreted as the population genotype frequencies for a different genetic locus (locus 2).
trans1	An ngeno1^2 by ngeno1 matrix of non-negative numbers whose rows sum to 1, where ngeno1 = length(geno_freq1). This matrix is usually generated by trans_monogenic or a similar helper function, and its elements are interpreted as genetic transmission probabilities for locus 1 (see trans_monogenic for more details). If trans1 has ngeno1 + 2 instead of ngeno1 columns, as could occur if it was generated by trans_monogenic with annotate = TRUE, then the first two columns will be deleted and trans1 will be converted to a matrix.
trans2	Similar to trans1 (above) but interpreted as the genetic transmission probabilities for locus 2.
annotate	A logical flag. When FALSE (the default), the function returns objects that can be used as the geno_freq and trans arguments of pedigree_loglikelihood. When TRUE, the function annotates these objects (and converts trans to a data frame) to make the output more easily understood by humans.

Details

This function combines the genotype frequencies and transition probabilities of two unlinked genetic loci that are in linkage equilibrium in a given population. Because the loci are unlinked, any person's genotypes at the two loci are conditionally independent given his or her parental genotypes, and because the loci are in linkage equilibrium, the genotypes at the two loci for a random person from the population are independent. This function uses these assumptions to calculate the population frequencies and transition probabilities for the joint genotypes of the two loci, where a joint genotype is just a pair consisting of a genotype at locus 1 and a genotype at locus 2. If the annotate option is set to FALSE then these frequencies and probabilities can be used in pedigree_loglikelihood to model the combined effect of the two loci on the phenotypes. By a repeated application of this function, more than two genetic loci can be included in the genetic model.

Value

A list with the following components:

geno_freq	A vector of strictly positive numbers (the joint genotype frequencies) that sum to 1, with genotype names added when annotate is TRUE
trans	Either a matrix of genetic transmission probabilities suitable to be used as the trans argument of pedigree_loglikelihood (if annotate is FALSE), or a data frame that is an annotated version of this matrix (if annotate is TRUE).

genotype_decoder
: A data frame giving the locus 1 and locus 2 genotypes that correspond to each joint genotype. In some cases, this could aid the user's calculation of the penet argument of pedigree_loglikelihood.

Examples

```
pa1 <- c(0.9, 0.1); names(pa1) <- c("-","+")
pa2 <- c(0.5, 0.5); names(pa2) <- c("A","a")
(geno_freq1 <- geno_freq_monogenic(pa1, TRUE))
(geno_freq2 <- geno_freq_monogenic(pa2, TRUE))
(trans1 <- trans_monogenic(2, TRUE))
(trans2 <- trans_monogenic(2))
(cl <- combine_loci(geno_freq1, geno_freq2, trans1, trans2, TRUE))
sum(cl$geno_freq)
apply(cl$trans[,-(1:2)], 1, sum)
```

---

dat_large *Simulated data on one family with approximately 10,000 members*

Description

A dataset giving the relationship structure of one large family and phenotypic data on the family members

Usage

```
dat_large
```

Format

A data frame with 10,002 rows (corresponding to persons) and the following 6 variables:

`indiv` an individual identifier for each person

`mother` the individual identifier of the person's mother

`father` the individual identifier of the person's father

`sex` the person's sex (1 = male, 2 = female)

`aff` the person's disease status (1 = case, 0 = control)

`age` the person's last known age in years (for controls) or age at diagnosis in years (for cases)

Source

Simulated

dat_seg *Sample Pedigree Data for Segregation Analyses*

Description

A data frame containing data on 20 families, suitable to be used as the `dat` argument of segregation_analysis.

Usage

```
dat_seg
```

Format

A data frame with 366 rows and 13 columns, with one row per person. The columns are described in detail in the `dat` argument of segregation_analysis but, in summary, the columns are:

family An identifier for each person's family.

indiv An individual identifier for each person.

mother The individual identifier of each person's mother.

father The individual identifier of each person's father.

sex The person's biological sex.

geno The person's genotype at the locus of interest, where the allele "+" is a rare genetic variant and "-" is the wildtype allele.

aff.col The person's colorectal cancer affected status, equal to `1` for colorectal cancer cases and `0` for non-cases.

age.col The person's age at diagnosis (for colorectal cancer cases) or last known age (for non-cases).

aff.end The person's endometrial cancer affected status, equal to `1` for endometrial cancer cases and `0` for non-cases.

age.end The person's age at diagnosis (for endometrial cancer cases) or last known age (for non-cases).

proband A binary flag equal to `1` for each family's proband (the first person in the family recruited into the study.

country The person's country of residence.

clinic A binary flag equal to `1` for families recruited via family cancer clinics or `0` for families recruited irrespective of the proband's family history of cancer.

Source

Simulated.

dat_small — *Simulated data on 10 families with approximately 100 members each*

Description

A dataset giving the relationship structure of 10 families and phenotypic data on the family members

Usage

```
dat_small
```

Format

A data frame with 1018 rows (corresponding to persons) and the following 7 variables:

`family` an identifier for each person's family

`indiv` an individual identifier for each person

`mother` the individual identifier of the person's mother

`father` the individual identifier of the person's father

`sex` the person's sex (1 = male, 2 = female)

`aff` the person's disease status (1 = case, 0 = control)

`age` the person's last known age in years (for controls) or age at diagnosis in years (for cases)

`geno` the person's genotype, or blank (`""`) if not known

Source

Simulated

fit_seg — *The Fitted Model from a Segregation Analysis*

Description

An object of class stats4::mle containing the maximum likelihood estimates from the segregation analysis in Examples of segregation_analysis.

Usage

```
fit_seg
```

Format

An object of class `mle` of length 1.

Source

Calculated by the not-run part of Examples in segregation_analysis.

geno_freq_monogenic *Calculate genotype frequencies from allele frequencies using Hardy-Weinberg equilibrium*

Description

A function to calculate the unphased genotype frequencies for a single autosomal genetic locus that has given allele frequencies and is at Hardy-Weinberg equilibrium (HWE).

Usage

```
geno_freq_monogenic(p_alleles, annotate = FALSE)
```

Arguments

p_alleles	A vector of strictly positive numbers that sum to 1, with p_alleles[i] interpreted as the allele frequency of the ith allele of the genetic locus. When annotate is TRUE, the names of the alleles will be taken to be names(p_alleles) or, if names(p_alleles) is NULL, to be 1:length(p_alleles).
annotate	A logical flag. When FALSE (the default), the function returns a vector suitable to be used as the geno_freq argument of pedigree_loglikelihood. When TRUE, the function adds a names attribute to this vector to indicate which genotype corresponds to which element.

Details

For a genetic locus at Hardy-Weinberg equilibrium, the population allele frequencies at the locus determine the population genotype frequencies; see Sections 1.2 and 1.3 of (Lange, 2002). Given a vector p_alleles containing the allele frequencies, this function returns the frequencies of the possible unphased genotypes, in a particular order that can be viewed by setting annotate to TRUE. If the alleles are named 1:length(p_alleles), so that p_alleles[i] is the frequency of allele i, then the unphased genotypes are named 1/1, 1/2, ... Note that if the output of this function is to be used as the geno_freq argument of pedigree_loglikelihood then the annotate option must be set to FALSE.

Value

A vector of strictly positive numbers (the genotype frequencies) that sum to 1, named with the genotype names if annotate is TRUE.

References

Lange K. Mathematical and Statistical Methods for Genetic Analysis (second edition). Springer, New York. 2002.

Examples

```
# Genotype frequencies for a biallelic locus at HWE and with a minor allele frequency of 10%
p_alleles <- c(0.9, 0.1)
geno_freq_monogenic(p_alleles, annotate = TRUE)

# Genotype frequencies for a triallelic locus at Hardy-Weinberg equilibrium
p_alleles <- c(0.85, 0.1, 0.05)
geno_freq_monogenic(p_alleles, annotate = TRUE)
sum(geno_freq_monogenic(p_alleles))
```

---

geno_freq_phased *Calculate phased genotype frequencies from allele frequencies, assuming Hardy-Weinberg equilibrium*

Description

A function to calculate the population frequencies of the phased genotypes at a single autosomal genetic locus that has given allele frequencies and is at Hardy-Weinberg equilibrium. Phased genotypes can be used to investigate parent-of-origin effects, e.g. see (van Vliet et al., 2011).

Usage

```
geno_freq_phased(p_alleles, annotate = FALSE)
```

Arguments

p_alleles	A vector of strictly positive numbers that sum to 1, with p_alleles[i] interpreted as the allele frequency of the ith allele of the genetic locus. When annotate is TRUE, the names of the alleles will be taken to be names(p_alleles) or, if names(p_alleles) is NULL, to be 1:length(p_alleles).
annotate	A logical flag. When FALSE (the default), the function returns a vector suitable to be used as the geno_freq argument of pedigree_loglikelihood. When TRUE, the function adds a names attribute to this vector to indicate which element corresponds to which phased genotype.

Details

For a genetic locus that is at Hardy-Weinberg equilibrium in a particular population, the population allele frequencies at the locus determine the population genotype frequencies; see Sections 1.2 and 1.3 of (Lange, 2002) for the unphased version of this law. When a genetic locus is at Hardy-Weinberg equilibrium, the maternal and paternal alleles of a random person from the population are independent. A phased genotype at a genetic locus is an ordered pair consisting of a maternal and paternal allele at the locus. So to any heterozygous unphased genotype, there are two corresponding phased genotypes, and these two phased genotypes have equal frequencies under Hardy-Weinberg equilibrium.

Given a vector p_alleles containing the allele frequencies, this function returns the frequencies of the possible phased genotypes, in a particular order that can be viewed by setting annotate to TRUE. If the alleles are named 1:length(p_alleles), so that p_alleles[i] is the frequency of allele i, then the phased genotypes are of the form 1|1, 1|2, ..., where a|b means the maternal allele is a and the paternal allele is b. Note that if the output of this function is to be used as the geno_freq argument of pedigree_loglikelihood then the annotate option must be set to FALSE.

Value

A vector of strictly positive numbers (the genotype frequencies) that sum to 1, named with the genotype names if annotate is TRUE.

References

Lange K. Mathematical and Statistical Methods for Genetic Analysis (second edition). Springer, New York. 2002.

van Vliet CM, Dowty JG, van Vliet JL, et al. Dependence of colorectal cancer risk on the parent-of-origin of mutations in DNA mismatch repair genes. Hum Mutat. 2011;32(2):207-212.

Examples

```
# Genotype frequencies for a biallelic locus at Hardy-Weinberg equilibrium
# and with a minor allele frequency of 10%
p_alleles <- c(0.9, 0.1)
geno_freq_phased(p_alleles, annotate = TRUE)

# Genotype frequencies for a triallelic locus at Hardy-Weinberg equilibrium
p_alleles <- c(0.85, 0.1, 0.05)
geno_freq_phased(p_alleles, annotate = TRUE)
sum(geno_freq_phased(p_alleles))
```

---

geno_freq_polygenic *Genotype frequencies for the hypergeometric polygenic model*

Description

A function to calculate the genotype frequencies for the hypergeometric polygenic model of (Cannings et al., 1978), see Section 8.9 of (Lange, 2002) for a nice description of this model.

Usage

```
geno_freq_polygenic(n_loci, annotate = FALSE)
```

Arguments

n_loci	A positive integer, interpreted as the number of biallelic genetic loci that contribute to the polygene. The polygene will have 2*n_loci + 1 genotypes, so n_loci is typically fairly small, e.g. 4.
annotate	A logical flag. When FALSE (the default), the function returns a vector suitable to be used as the geno_freq argument of pedigree_loglikelihood. When TRUE, the function adds a names attribute to this vector to indicate which element corresponds to which genotype.

Details

The hypergeometric polygenic model (Cannings et al., 1978; Lange, 2002) is a computationally feasible genetic model that approximates the combined effect of a given number (n_loci) of unlinked biallelic genetic loci. This model is often used to model the effect of such loci on a trait when the alleles at these loci either increase the trait by a certain, locus-independent amount (if a 'positive' allele) or decrease the trait by the same amount (if a 'negative' allele), with 'positive' and 'negative' alleles equally likely at each locus. In this case, the only relevant aspect of the 3 ^ n_loci possible joint genotypes is the total number of 'positive' alleles, so the possible genotypes of the hypergeometric polygenic model are taken to be 0:(2*n_loci). The transmission probabilities and genotype frequencies of the hypergeometric polygenic model approximate these quantities for the combination of the n_loci biallelic genetic loci described above. Under this model, the polygenic genotype for each person is approximately normally distributed, and these genotypes are correlated within families with correlation coefficients (in non-inbred families) equal to the kinship coefficients (Lange, 2002).

Setting annotate to TRUE names each element of the output vector with the corresponding genotype. The annotate option must be set to FALSE if the output of this function is to be used as the geno_freq argument of pedigree_loglikelihood.

Value

A vector of strictly positive numbers (the genotype frequencies) that sum to 1, named with the genotype names if annotate is TRUE.

References

Cannings C, Thompson E, Skolnick M. Probability functions on complex pedigrees. Advances in Applied Probability, 1978;10(1):26-61.

Lange K. Mathematical and Statistical Methods for Genetic Analysis (second edition). Springer, New York. 2002.

Examples

```
geno_freq_polygenic(4, annotate = TRUE)
sum(geno_freq_polygenic(4))
```

`genotype_probabilities` *Calculate genotype probabilities for a target person*

Description

For a chosen individual within a specified family, calculate the person's conditional genotype probabilities, given the family's phenotypes and relationship structure

Usage

```
genotype_probabilities(target, fam, geno_freq, trans, penet, monozyg = NULL)
```

Arguments

`target`	The individual identifier (an element of `fam$indiv`) of the person in the pedigree `fam` whose genotype probabilities are being sought.
`fam`	A data frame specifying the family's relationship structure, with rows corresponding to people and columns corresponding to the following variables (other variables can be included but will be ignored), which will be coerced to `character` type: • `indiv`, an identifier for each individual person, with no duplicates in `fam`. • `mother`, the individual identifier of each person's mother, or missing (`NA`) for founders. • `father`, the individual identifier of each person's father, or missing (`NA`) for founders.
`geno_freq`	A vector of strictly positive numbers that sum to `1`. If the possible genotypes of the underlying genetic model are `1:length(geno_freq)` then `geno_freq[j]` is interpreted as the population frequency of genotype `j`. For certain genetic models that often occur in applications, these genotype frequencies can be calculated by `geno_freq_monogenic`, `geno_freq_phased`, etc.
`trans`	An `ngeno^2` by `ngeno` matrix of non-negative numbers whose rows all sum to `1`, where `ngeno = length(geno_freq)` is the number of possible genotypes. The rows of `trans` correspond to joint parental genotypes and the columns correspond to offspring genotypes. If the possible genotypes are `1:length(geno_freq)` then the element `trans[ngeno * gm + gf - ngeno, go]` is interpreted as the conditional probability that a person has genotype `go`, given that his or her biological mother and father have genotypes `gm` and `gf`, respectively. For certain genetic models that often occur in applications, this transmission matrix can be calculated by `trans_monogenic`, `trans_phased`, etc.
`penet`	An `nrow(fam)` by `length(geno_freq)` matrix of non-negative numbers. The element `penet[i,j]` is interpreted as the conditional probability (or probability density) of the phenotype of the person corresponding to row `i` of `fam`, given that his or her genotype is `j` (where the possible genotypes are `1:length(geno_freq)`). Note that genotype data can be incorporated into `penet` by regarding observed

genotypes as part of the phenotype, i.e. by regarding observed genotypes as (possibly noisy) measurements of the underlying true genotypes. For example, if the observed genotype of person `i` is `1` (and if genotype measurement error is negligible) then `penet[i,j]` should be `0` for `j != 1` and `penet[i,1]` should be the same as if person `i` were ungenotyped.

`monozyg` An optional list that can be used to specify genetically identical persons, such as monozygotic twins, monozygotic triplets, a monozygotic pair within a set of dizygotic triplets, etc. Each element of the list should be a vector containing the individual identifiers of a group of genetically identical persons, e.g. if `fam` contains a set of monozygotic twins (and no other genetically identical persons) then `monozyg` will be a list with one element, and that element will be a vector of length two containing the individual identifiers of the twins. The order of the list and the orders of its elements do not affect the output of the function. Each group of genetically identical persons should contain two or more persons, the groups should not overlap, and all persons in each group must have the same (non-missing) parents.

Details

The genotype probabilities are calculated by essentially the same algorithm as `pedigree_loglikelihood`; see there for details. The genotype probabilities only depend on the connected component of the pedigree that contains `target`, so the function first restricts `fam` and `penet` to the rows corresponding to this connected component. For example, if `fam` is the union of two unrelated families then this function will restrict to the subfamily containing `target` before performing the calculation.

Value

A vector of length `length(geno_freq)`, whose `j`th element is the conditional probability that the target person has genotype `j`, given the family's relationship structure and phenotypes. A vector of `NA`s will be returned if a row of `penet` consists entirely of zeroes or if the pedigree is impossible for any other reason (after restricting `fam` and `penet` to the connected component of the pedigree containing `target`).

Examples

```
# Read in some sample data
data("dat_small", "penet_small")
str(dat_small)
str(penet_small)

# Calculate the genotype probabilities for individual "ora008" in the family "ora"
w <- which(dat_small$family == "ora")
fam <- dat_small[w, -1]
penet <- penet_small[w, ]
monozyg <- list(c("ora024", "ora027"))  # ora024 and ora027 are identical twins
trans <- trans_monogenic(2)
geno_freq <- geno_freq_monogenic(p_alleles = c(0.9, 0.1))
genotype_probabilities(target = "ora008", fam, geno_freq, trans, penet, monozyg)
```

inc_seg *Population Incidences for a Segregation Analysis*

Description

A data frame containing age-, sex- and country-specific annual colorectal and endometrial cancer incidences for the general population in 1998-2002.

Usage

```
inc_seg
```

Format

A data frame with 81 rows (corresponding to ages `0` to `80` years) and 9 columns. All variables except `Age` contain age-specific cancer annual incidences, and these eight columns have names of the form `<disease>.<country>.<sex>`, where: `disease` ranges over `col` and `end`, for colorectal and endometrial cancer; `country` ranges over `Aus` and `USA`, for Australia and the United States; and `sex` ranges over `M` and `F`, for male and female. In summary, the columns are:

Age Ages ranging from `0` to `80` years. Note that row `i` corresponds to age `i-1`.

col.Aus.M Colorectal cancer population incidences for male Australian residents.

col.Aus.F Colorectal cancer population incidences for female Australian residents.

col.USA.M Colorectal cancer population incidences for male United States residents.

col.USA.F Colorectal cancer population incidences for female United States residents.

end.Aus.M Endometrial cancer population incidences for male Australian residents (identically zero).

end.Aus.F Endometrial cancer population incidences for female Australian residents.

end.USA.M Endometrial cancer population incidences for male United States residents (identically zero).

end.USA.F Endometrial cancer population incidences for female United States residents.

Source

Curado MP, Edwards B, Shin HR, Storm H, Ferlay J, Heanue M, Boyle P (editors). *Cancer Incidence in Five Continents, Vol. IX*. IARC Scientific Publications, No. **160**. 2007.

`pedigree_loglikelihood` *Calculate the log-likelihoods of pedigrees*

Description

For one or more pedigrees, this function calculates the natural logarithm of the pedigree likelihood that is on page 117 of (Lange, 2002), given inputs that correspond to the terms in this formula.

Usage

```
pedigree_loglikelihood(
  dat,
  geno_freq,
  trans,
  penet,
  monozyg = NULL,
  sum_loglik = TRUE,
  ncores = 1,
  load_balancing = TRUE
)
```

Arguments

`dat`	A data frame with rows corresponding to people and columns corresponding to the following variables (other variables can be included but will be ignored), which will be coerced to `character` type: • `family` (optional), an identifier for each person's family, constant within families. If this variable is not supplied then `dat` will be treated as a single pedigree. • `indiv`, an individual identifier for each person. If there are any duplicated identifiers in the dataset then the family and an underscore (_) will be prepended to all identifiers, and if any duplicates remain after this then the function will stop executing, with an error message. • `mother`, the individual identifier of each person's mother, or missing (`NA`) for founders. • `father`, the individual identifier of each person's father, or missing (`NA`) for founders.
`geno_freq`	A vector of strictly positive numbers that sum to 1. If the possible genotypes of the underlying genetic model are `1:length(geno_freq)` then `geno_freq[j]` is interpreted as the population frequency of genotype `j`, so `geno_freq` is essentially the function `Prior` in the pedigree likelihood on page 117 of (Lange, 2002). For certain genetic models that often occur in applications, these genotype frequencies can be calculated by `geno_freq_monogenic`, `geno_freq_phased`, etc.

`trans` An `ngeno^2` by `ngeno` matrix of non-negative numbers whose rows all sum to `1`, where `ngeno = length(geno_freq)` is the number of possible genotypes. The rows of `trans` correspond to joint parental genotypes and the columns correspond to offspring genotypes. If the possible genotypes are `1:length(geno_freq)` then the element `trans[ngeno * gm + gf - ngeno, go]` is interpreted as the conditional probability that a person has genotype `go`, given that his or her biological mother and father have genotypes `gm` and `gf`, respectively. So `trans` is essentially the transmission function `Tran` on page 117 of (Lange, 2002). For certain genetic models that often occur in applications, this transmission matrix can be calculated by `trans_monogenic`, `trans_phased`, etc.

`penet` An `nrow(dat)` by `length(geno_freq)` matrix of non-negative numbers. The element `penet[i,j]` is interpreted as the conditional probability (or probability density) of the phenotype of the person corresponding to row `i` of `dat`, given that his or her genotype is `j` (where the possible genotypes are `1:length(geno_freq)`). Therefore, `penet` is essentially the penetrance function `Pen` on page 117 of (Lange, 2002). If any row of `penet` consists entirely of zeroes then the likelihood is `0`, so the returned log-likelihood will be `-Inf`. Note that genotype data can be incorporated into `penet` by regarding observed genotypes as part of the phenotype, i.e. by regarding observed genotypes as (possibly noisy) measurements of the underlying true genotypes. For example, if the observed genotype of person `i` is `1` (and if genotype measurement error is negligible) then `penet[i,j]` should be `0` for `j != 1` and `penet[i,1]` should be the same as if person `i` were ungenotyped.

`monozyg` An optional list that can be used to specify genetically identical persons, such as monozygotic twins, monozygotic triplets, a monozygotic pair within a set of dizygotic triplets, etc. Each element of the list should be a vector containing the individual identifiers of a group of genetically identical persons, e.g. if `dat` contains six sets of monozygotic twins and one set of monozygotic triplets then `monozyg` will be a list with seven elements, one element a vector of length three and the other six elements all vectors of length two. The order of the list and the orders within its elements do not affect the output of the function. Each group of genetically identical persons should contain two or more persons, the groups should not overlap, and all persons in each group must have the same (non-missing) parents.

`sum_loglik` A logical flag. Return a named vector giving the log-likelihood of each family if `sum_loglik` is `FALSE`, or return the sum of these log-likelihoods if `sum_loglik` is `TRUE` (the default).

`ncores` The number of cores to be used, with `ncores = 1` (the default) corresponding to non-parallel computing. When `ncores > 1`, the `parallel` package is used to parallelize the calculation by dividing the pedigrees among the different cores.

`load_balancing` A logical flag. When `ncores > 1`, parallelization is achieved either with the function `parallel::parLapply` (if `load_balancing` is `FALSE`) or with the load-balancing function `parallel::parLapplyLB` (if `load_balancing` is `TRUE`, the default). The load-balancing version will usually, but not always, be faster.

Details

This function provides a fast and general implementation of the Elston-Stewart algorithm to calculate the log-likelihoods of potentially large and complex pedigrees. General references for the Elston-Stewart algorithm are (Elston & Stewart, 1971), (Lange & Elston, 1975) and (Cannings et al., 1978).

Each family within `dat` should be a complete pedigree, meaning that each person should either have both parental identifiers missing (if a founder) or both non-missing (if a non-founder), and each (non-missing) mother or father should have a corresponding row of `dat`.

Observed genotypes should be incorporated into `penet`, as described above.

The function can handle pedigree loops, such as those caused by inbreeding or by two sisters having children with two brothers from an unrelated family (see (Totir et al., 2009) for a precise definition), though pedigrees with more than a few loops could greatly reduce the speed of the calculation.

In `geno_freq`, `trans` and `penet`, the order of the possible genotypes must match, in the sense that the genotype that corresponds to element `j` of `geno_freq` must also correspond to column `j` of `trans` and `penet`, for each `j` in `1:length(geno_freq)`.

Sex-specific genetics, such as X-linked genes or genetic loci with sex-specific recombination fractions, can be modeled by letting genotypes `1:nm` be the possible male genotypes and letting `(nm+1):(nm+nf)` be the possible female genotypes, where `nm` and `nf` are the number of possible genotypes for males and females, respectively. Then, for example, `penet[i,j]` will be `0` if `j %in% 1:nm` and row `i` of `dat` corresponds to a female, and `penet[i,j]` will be `0` if `j %in% (nm+1):(nm+nf)` and row `i` of `dat` corresponds to a male.

Value

Either a named vector giving the log-likelihood of each family or the sum of these log-likelihoods, depending on `sum_loglik` (see above).

References

Cannings C, Thompson E, Skolnick M. Probability functions on complex pedigrees. Advances in Applied Probability, 1978;10(1):26-61.

Elston RC, Stewart J. A general model for the genetic analysis of pedigree data. Hum Hered. 1971;21(6):523-542.

Lange K. Mathematical and Statistical Methods for Genetic Analysis (second edition). Springer, New York. 2002.

Lange K, Elston RC. Extensions to pedigree analysis I. Likelihood calculations for simple and complex pedigrees. Hum Hered. 1975;25(2):95-105.

Totir LR, Fernando RL, Abraham J. An efficient algorithm to compute marginal posterior genotype probabilities for every member of a pedigree with loops. Genet Sel Evol. 2009;41(1):52.

Examples

```
# Load pedigree files and penetrance matrices
data("dat_small", "penet_small", "dat_large", "penet_large")

# Settings for a single biallelic locus in Hardy-Weinberg equilibrium
```

```
# and with a minor allele frequency of 10%
geno_freq <- geno_freq_monogenic(c(0.9, 0.1))
trans <- trans_monogenic(2)

# In dat_small, ora024 and ora027 are identical twins, and so are aey063 and aey064
monozyg_small <- list(c("ora024", "ora027"), c("aey063", "aey064"))

# Calculate the log-likelihoods for 10 families, each with approximately
# 100 family members
pedigree_loglikelihood(
  dat_small, geno_freq, trans, penet_small, monozyg_small, sum_loglik = FALSE, ncores = 2
)

# Calculate the log-likelihood for one family with approximately 10,000 family members
# Note:  this calculation should take less than a minute on a standard desktop computer
# Note:  parallelization would achieve nothing here because there is only one family
str(dat_large)

system.time(
  ll <- pedigree_loglikelihood(dat_large, geno_freq, trans, penet_large)
)
ll
```

`penet_large` *A penetrance matrix relating the phenotypes in* `dat_large` *to three genotypes*

Description

A matrix relating the phenotypes of `dat_large` to the three unphased genotypes of a single biallelic, autosomal genetic locus. The element `penet_large[i,j]` is the conditional probability of the phenotypes (i.e. `sex`, `aff` and `age`) of the person in row `i` of `dat_large`, given that his or her genotype is `j` (here labeling the genotypes as `1`, `2`, `3`, where genotype `2` is the heterozygous genotype).

Usage

```
penet_large
```

Format

A matrix with 10,002 rows (corresponding to persons) and 3 columns (corresponding to genotypes).

Source

Simulated

penet_small — *A penetrance matrix relating the phenotypes of* dat_small *to three genotypes*

Description

A matrix relating the phenotypes of dat_small to the three unphased genotypes of a single biallelic, autosomal genetic locus. The element penet_small[i,j] is the conditional probability of the phenotypes (i.e. sex, aff and age) of the person in row i of dat_small, given that his or her genotype is j (here labeling the genotypes as 1, 2, 3, where genotype 2 is the heterozygous genotype).

Usage

```
penet_small
```

Format

A matrix with 1018 rows (corresponding to persons) and 3 columns (corresponding to genotypes).

Source

Simulated

plot_pedigree — *Plot Pedigree Pictures*

Description

Plots the pedigree picture (family tree) for the supplied family, with options that allow the user to easily plot common data on each person. This is wrapper for the harder to use (but more flexible) pedigree plotting function plot_pedigree_adv.

Usage

```
plot_pedigree(
  fam,
  aff.vars = NULL,
  aff.cols = rainbow(length(aff.vars)),
  under.text = substr(fam$indiv, nchar(fam$indiv) - 2, nchar(fam$indiv)),
  red = NULL,
  green = NULL,
  blue = NULL,
  arrow = NULL,
  box = NULL,
  box.col = "red",
```

```
  incircle = NULL,
  incircle.col = "red",
  textcex = 1,
  indiv.box.dim = c(2, 2, 2, 4),
  lwd = 1,
  title.txt = "",
  ntries = 20,
  rnd = NULL,
  outfile = NULL
)
```

Arguments

fam	A data frame with rows corresponding to people and columns corresponding to the following variables: • indiv: the individual identifier for each person, which should be of character type. • mother: the individual identifier of each person's mother, or missing (NA) for founders. • father: the individual identifier of each person's father, or missing (NA) for founders. • sex: a character variable determining the shape of the symbol for each person. This shape will be a square (if sex is "M"), a circle (if sex is "F") or a diamond (otherwise). • ...: any other variables which you would like to use when displaying the symbol and data for each person.
aff.vars	A subset of names(fam) specifying logical or numeric variables of fam that will be used to determine how each person's symbol will be coloured in. Each variable listed in aff.vars corresponds to a fixed segment of the symbol, in clockwise order from 12 o'clock. The ith segment, corresponding to aff.vars[i], will be coloured aff.cols[i] for people with fam[, aff.vars[i]] either TRUE or non-zero, otherwise the segment will be coloured white. See Examples, below.
aff.cols	A character vector the same length as aff.vars that contains the possible colours of the segments of each person's symbol. See aff.vars, above.
under.text	A character vector of length nrow(fam) containing text to be printed below each person's symbol.
red	A numeric vector of length nrow(fam) with values between 0 and 1 (inclusive), which specifies the amount of red in the colour of a highlighted rectangle covering each person's symbol and data. Having red equal to NULL (the default) is equivalent to setting red equal to 0 for all family members. If red, green and blue are all zero then no highlighted box will be displayed. See Examples, below.
green	A numeric vector of length nrow(fam), the same as red (above) but specifying the amount of green in the colour of each person's highlighted rectangle.
blue	A numeric vector of length nrow(fam), the same as red (above) but specifying the amount of blue in the colour of each person's highlighted rectangle.

`arrow` A `logical` or `numeric` vector of length `nrow(fam)`. The plot will have small black arrows pointing at each person for whom `arrow` is either `TRUE` or non-zero.

`box` A `logical` or numeric vector of length `nrow(fam)`, determining which people will be displayed with a box drawn around their symbol and data.

`box.col` A character string specifying the colour of the box displayed for people with `box` either `TRUE` or non-zero.

`incircle` A `logical` or `numeric` vector of length `nrow(fam)`, determining which people will have a small filled circle added to the centre of their symbols, with the fill colour determined by `incircle.col`.

`incircle.col` A character string specifying the fill colour of the small circle displayed for people with `incircle` either `TRUE` or non-zero.

`textcex` A positive number multiplying the size of the text in `under.text`. This argument `textcex` will be passed to the `text` function via its argument `cex`.

`indiv.box.dim` A vector of four positive numbers, determining the dimensions and placement of a rectangle containing an individual person's symbol and his or her displayed data. This vector should be `c(left, right, top, bottom)`, where each element says how far the relevant side of the rectangle is from the centre of each person's symbol.

`lwd` A positive integer giving the width of all plotted lines.

`title.txt` A character string containing text that will be used as the title of the plot, which by default is equal to `""` (no title).

`ntries` A positive integer giving the number of random layouts that the function will try before selecting the most compact and aesthetically pleasing layout for the plotted family. Larger values will generally give nicer plots but will have longer compute times.

`rnd` A positive integer which will be set as the random seed before trying random layouts for the pedigree plot. If `rnd` is `NULL` (the default) then no random seed is set within the function. Supplying a random seed is the easiest way to make the outputted plot repeatable, i.e. deterministic.

`outfile` A character string containing the name of a PDF file in which you would like to store the pedigree plot. For example, setting `outfile = "myfam.pdf"` will cause the plot that is printed on the screen to also be stored in a file `myfam.pdf` in the current working directory. If `outfile` is `NULL` (the default) then the plot will only be printed to the screen.

Value

A data frame with rows corresponding to all people who were plotted (some people might be omitted, e.g. if the pedigree is not connected) and columns corresponding to the following variables for each person:

- `indiv`: the individual identifier of the person.
- `x`: the x-coordinate of the centre of the person's plotted symbol.
- `y`: the y-coordinate of the centre of the person's plotted symbol.

Examples

```
# Read in a sample family to plot and add some dummy variables
dat <- clipp::dat_seg
fam <- dat[dat$family == "bac", ]
set.seed(10)
fam$blood <- sample(c(0,1), nrow(fam), replace=TRUE)
fam$aff.oth <- sample(c(0,1), nrow(fam), replace=TRUE)

# Plot the family
utext <- substr(fam$indiv, nchar(fam$indiv)-1, nchar(fam$indiv))
utext <- paste(utext, fam$geno, sep="\n")
pp <- plot_pedigree(fam, aff.vars=c("aff.col", "aff.oth", "aff.end"),
                    aff.cols=c("blue", "green", "purple"),
                    red=is.na(fam$father) & fam$sex=="F",
                    blue=is.na(fam$father) & fam$sex=="M",
                    arrow=fam$proband, incircle=fam$blood,
                    lwd=1, under.text=utext,
                    textcex=0.8, ntries=100, rnd=1)
head(pp)

# Use thicker lines and gold highlights for all founders
hue <- as.numeric(col2rgb("gold"))/255
r <- hue[1] * as.numeric(is.na(fam$father))
g <- hue[2] * as.numeric(is.na(fam$father))
b <- hue[3] * as.numeric(is.na(fam$father))
pp <- plot_pedigree(fam, aff.vars=c("aff.col", "aff.oth", "aff.end"),
                    aff.cols=c("blue", "green", "purple"),
                    red=r, green=g, blue=b,
                    arrow=fam$proband, incircle=fam$blood,
                    lwd=2, under.text=utext,
                    textcex=0.8, ntries=100, rnd=1)
head(pp)
```

`plot_pedigree_adv` *Plot Pedigree Pictures (Advanced)*

Description

Plots the pedigree picture (family tree) for the supplied family, with data on each person plotted in a custom way via an function specified by the user. This is a more flexible but harder to use version of the pedigree plotting function plot_pedigree.

Usage

```
plot_pedigree_adv(
  fam,
  indiv.plot.fn,
  indiv.box.dim = c(2, 2, 2, 4),
```

```
  lwd = 1,
  title.txt = "",
  ntries = 20,
  rnd = NULL,
  outfile = NULL
)
```

Arguments

fam	A data frame with rows corresponding to people and columns corresponding to the following variables: • indiv: the individual identifier for each person, which should be of character type. • mother: the individual identifier of each person's mother, or missing (NA) for founders. • father: the individual identifier of each person's father, or missing (NA) for founders. • ...: any other variables which you would like to use when displaying the symbol and data for each person using indiv.plot.fn.
indiv.plot.fn	A function for plotting each person's symbol and any data of interest. This function should have arguments ID, x and y, where ID is the person's individual identifier (an element of fam$indiv), and x and y are the Cartesian co-ordinates of the centre of the person's symbol. The plot function can display any data in fam, though this displayed data should always fit within a box determined by x, y and indiv.box.dim (see below). For examples of this function, see Examples (below) and the code for plot_pedigree (which defines an indiv.plot.fn and then calls plot_pedigree_adv).
indiv.box.dim	A vector of four positive numbers, determining the dimensions and placement of the rectangle containing an individual person's symbol and his or her displayed data. This vector should be c(left, right, top, bottom), where each element says how far the relevant side of the rectangle is from a central point (the point with co-ordinates x and y in indiv.plot.fn).
lwd	A positive integer determining the width of the lines joining the symbols of parents and their children.
title.txt	Text that will be used as the title of the plot, equal to "" (no title) by default.
ntries	A positive integer giving the number of random layouts that the function will try before selecting the most compact and aesthetically pleasing layout for the plotted family. Larger values will generally give nicer plots but will have longer compute times.
rnd	A positive integer which will be set as the random seed before trying random layouts for the pedigree plot. If rnd is NULL (the default) then no random seed is set within the function. Supplying a random seed is the easiest way to make the outputted plot repeatable, i.e. deterministic.
outfile	A character string containing the name of a PDF file in which you would like to store the pedigree plot. For example, setting outfile = "myfam.pdf" will cause the plot that is printed on the screen to also be stored in a file myfam.pdf

in the current working directory. If `outfile` is `NULL` (the default) then the plot will only be printed to the screen.

Value

A data frame with rows corresponding to all people who were plotted (some people might be omitted, e.g. if the pedigree is not connected) and columns corresponding to the following variables for each person:

- `indiv`: the individual identifier of the person.
- `x`: the x-coordinate of the centre of the person's plotted symbol.
- `y`: the y-coordinate of the centre of the person's plotted symbol.

Examples

```
# Read in a sample family to plot
dat <- clipp::dat_seg
fam <- dat[dat$family == "bac", ]

# A function to plot the symbol for a person's given sex at point (x0, y0)
plot.symbol <- function(x0, y0, sex, col=NA, lwd) {
  if (sex == "F")  theta <- seq(0, 2*pi, length.out=101)
  if (sex == "M")  theta <- seq(0, 2*pi, length.out=9)
  if (!sex %in% c("M", "F"))  theta <- seq(0, 2*pi, length.out=5)
  r <- rep(1, length(theta))
  if (sex == "M")  r <- r * c(rep(c(1,sqrt(2)), 4), 1)
  polygon(x0 + r*cos(theta), y0 + r*sin(theta), col=col, lwd=lwd)
}

indiv.plot.fn <- function(ID, x, y) {
  # A function to plot at the point (x,y) the symbol and data of the person
  # in `fam` with individual identifier `ID`.

  # Choose a colour for the person's symbol, based on affected status
  k <- which(fam$indiv==ID)
  symcol <- "white"
  if (fam$aff.col[k] == 1)  symcol <- "blue"
  if (fam$aff.end[k] == 1)  symcol <- "red"
  if (fam$aff.col[k] == 1 & fam$aff.end[k] == 1)  symcol <- "purple"

  # Plot the current person's symbol
  plot.symbol(x, y, fam$sex[k], symcol, lwd=1)

  # Print the last 2 digits of the person's individual identifier under the symbol
  utext <- substr(ID, nchar(ID)-1, nchar(ID))
  text(x, y-1.2, labels=utext, adj=c(0.5, 1))

}

# Plot the pedigree
pp <- plot_pedigree_adv(fam, indiv.plot.fn, ntries=200, rnd=1)
head(pp)
```

segregation_analysis *Conduct a Segregation Analysis to Estimate Genetic Hazard Ratios*

Description

This function calculates maximum likelihood estimates for arbitrary hazard ratio parameters, using a segregation analysis that incorporates a parametric survival analysis. This survival analysis is based on known population incidences for the diseases of interest, rather than Poisson or other classical distributions. Multiple diseases can be modeled simultaneously, which is often advantageous when a genetic variant increases the risk of several diseases (e.g. Lynch syndrome increasing the risks of colorectal and endometrial cancer) since disease status gives probabilistic information about carrier status. Internally, this function calculates the corresponding penetrance matrix for any value of the unknown parameters, then uses pedigree_loglikelihood to calculate the likelihood function. It then calls the function mle from the stats4 base package to produce the maximum likelihood estimates and their standard errors.

Usage

```
segregation_analysis(
  theta.start,
  dat,
  geno_freq,
  trans,
  calc_inc,
  calc_HR,
  ncores = 1,
  monozyg = NULL,
  compat_genos = NULL,
  asc = NULL,
  verbose = FALSE,
  centre = TRUE
)
```

Arguments

theta.start	A named numeric vector, giving the starting values and the names of the parameters to be estimated using the method of maximum likelihood.
dat	A data frame with rows corresponding to people and columns corresponding to the following variables: • family: an identifier for each person's family, constant within families. This variable will be coerced to character type. • indiv: an individual identifier for each person, which should be unique across dat. Coerced to character type. • mother: the individual identifier of each person's mother, or missing (NA) for founders. Coerced to character type.

- `father`: the individual identifier of each person's father, or missing (`NA`) for founders. Coerced to `character` type.
- `aff.<disease>`: numeric variables giving the affected statuses of each disease included in the analysis. Each variable should be `0` for people without the given disease and `1` for people with the disease, with no missing values. In Examples below, the diseases are `col` and `end` so the corresponding affected statuses are columns `aff.col` and `aff.end` of dat_seg.
- `age.<disease>`: numeric variables giving, for each disease, either the age at diagnosis of the disease (for cases, i.e. people with affected status equal to `1`) or the last known age (for controls). Each variable should take values in `0:max_age` with no missing values, where `max_age` is the maximum age in the output of `calc_HR` and `calc_inc`. In Examples below, the diseases are `col` and `end` so the corresponding age variables are columns `age.col` and `age.end` of dat_seg.
- `geno` (optional): A variable coerced to character type which contains genotype data. This variable will be ignored if `compat_genos` is null (the default). Missing values (`NA`) and the empty string (`""`) are interpreted as no genetic testing. See `compat_genos` below for more information.
- `...`: demographic or other variables required by `calc_inc` or `calc_HR` as arguments, e.g. `sex` and `country` in Examples, below. None of these variables should be named `theta`, since `theta` is a special argument of `calc_HR`.

`geno_freq` A vector of strictly positive numbers that sum to `1`, interpreted as the population frequencies of all possible genotypes. See the `geno_freq` argument of pedigree_loglikelihood for more information.

`trans` A matrix of non-negative numbers whose rows all sum to `1`, containing the conditional probabilities for a person's genotype given his or her parental genotypes. See the `trans` argument of pedigree_loglikelihood for more information.

`calc_inc` A function that inputs demographic or other variables for any person in `dat` and returns the relevant population incidences for that person for all diseases included in the analysis. The demographic variables can be any subset of the variables in `dat`, including the empty set (i.e. no demographic variables). The output of this function should be a named list, with each element corresponding to a given disease (even if there is only one disease) and each element of the list being a numeric vector containing the annual population incidences for the disease at ages `0, 1, 2, ...`. The maximum age should be the same for all diseases and for all people in `dat`. See Examples, below.

`calc_HR` A function that takes arguments `theta` and the demographic or other variables for any person in `dat` and returns the relevant disease-, age- and genotype-specific hazard ratios for that person. Here, `theta` is a named numeric vector with the same length and names as `theta.start`, and the demographic variables can be any subset of the variables in `dat`. The purpose of this function is to translate the arbitrary, user-defined parameters `theta`, which are to be estimated, into hazard ratios in a standard form. The output of this function should be a named list, with names exactly matching the names of the list outputted by `calc_inc` and each element (corresponding to a given disease) being a numeric matrix containing the hazard ratios at ages `0, 1, 2, ...` (in rows `1, 2, 3, ...`)

for each possible genotype (columns), as in Examples, below. The maximum age should be the same as in the output of `calc_inc` and the columns should correspond to genotypes in the same order as the elements of `geno_freq`. The hazard ratios can be with respect to a given genotype (e.g. non-carriers) or the population, depending on `centre` (see below).

`ncores` A positive integer specifying the number of cores to be used in the likelihood calculation, with `ncores = 1` (the default) corresponding to non-parallel computing.

`monozyg` An optional list that can be used to specify genetically identical persons, such as monozygotic twins or monozygotic triplets. Each element of the list should be a vector containing the individual identifiers of a group of genetically identical persons. See Examples in pedigree_loglikelihood. The order of the list and the order of vectors within its elements do not matter.

`compat_genos` A named list whose names range over the possible values of `dat$geno` (except `NA` and `""`) and whose elements are integer vectors. This list provides a link between the arbitrary, user-defined genotype data in `dat$geno` and the genotypes of the underlying genetic model defined by `geno_freq` and `trans`. Each element of `compat_genos` should be a subset of `1:length(geno_freq)` specifying the genotypes of the genetic model that are compatible with the value of `dat$geno` that the element of `compat_genos` is named after. For example, if the underlying model is based on phased genotypes but `dat$geno` only specifies unphased genotypes then a person with `dat$geno` equal to the heterozygote genotype `1/2` must have phased genotype `1|2` or `2|1`, so `compat_genos[["1/2"]]` should be `2:3` if the elements of `geno_freq` correspond (in order) to phased genotypes `1|1`, `1|2`, `2|1`, `2|2`. See Examples, below, for another example.

`asc` An optional data frame with the same columns as `dat`, used to condition the likelihood on ascertainment criteria to avoid ascertainment bias in the estimates of the hazard ratio parameters `theta` (a critical issue in penetrance estimation). If `asc` is supplied, the log-likelihood for `asc` will be subtracted from that of `dat`, then the resulting, conditional log-likelihood (i.e. the log-likelihood for the data conditioned on the ascertainment criteria) will be optimised to produce maximum likelihood estimates. See Examples below for conditioning on a combination of population- and clinic-based ascertainment.

`verbose` A logical variable determining how much information is printed to the screen while performing the segregation analysis, with `FALSE` (the default) corresponding to no intermediate information.

`centre` If `TRUE` (the default) then the baseline incidences will be chosen so that the overall incidence of disease is equal to the relevant population incidence. Here, the overall incidence of disease is the incidence in the population, unconditional on genotype (approximately equal to the weighted average of the genotype-specific incidences, with weights given by `geno_freq`) and the baseline incidence is the denominator in the definition of hazard ratio, i.e. the denominator of the age-specific incidence of disease for people with a given genotype divided by the age-specific incidence of disease in a reference group. Usually, the reference group is the set of non-carriers of a genetic variant, whose precise disease incidences are not known. In this and many other cases, the baseline incidences can be determined by the condition that the overall disease incidences should

equal the population incidences (if the variant is very rare then the resulting incidences for non-carriers will be approximately equal to the population incidences). However, in rare cases, hazard ratios are defined with respect to the population (rather than non-carriers), in which case `centre` should be set to `FALSE`. Though in that case, the overall incidences of disease will usually not be exactly equal to the supplied population incidences (and if they are equal then the results will be the same as when `centre` is `TRUE`).

Value

An object of class stats4::mle from the stats4 base package containing the maximum likelihood estimates and their standard errors. See Examples, below.

Examples

```
# Perform a segregation analysis to estimate the risks of
# colorectal cancer ("col") and endometrial cancer ("end")
# for carriers of a rare genetic variant.

# Define the genetic model -- an autosomal, biallelic locus in Hardy-Weinberg
# equilibrium and with a minor allele frequency of 1%.  The elements of geno_freq
# correspond (in order) to non-carriers of the rare genetic variant,
# heterozygote carriers and homozygote carriers
afreq <- 0.01
geno_freq <- geno_freq_monogenic(c(1-afreq, afreq))
trans <- trans_monogenic(2)

# Read in the annual population incidences for ages 0 to 80 years
popinc <- clipp::inc_seg
head(popinc)   # Row i of popinc corresponds to age i-1

# Read in the pedigree data
dat <- clipp::dat_seg
head(dat)

# Starting values of the hazard ratio parameters to be estimated
theta.start <- c(4, 2, 4, 2, 3)
names(theta.start) <- c("col.m.40", "col.m.60", "col.f.40", "col.f.60", "end.f")

# A function to return the relevant population incidences for the diseases
# of interest ("col" and "end") for a person with given demographic characteristics
calc_inc <- function(sex, country) {
  sc <- paste(country, sex, sep=".")
  inc.col <- popinc[, paste0("col.", sc)]
  inc.end <- popinc[, paste0("end.", sc)]
  list(col=inc.col, end=inc.end)
}
calc_inc("M", "Aus")   # Population incidences for male Australian residents for ages 0:80

# A function to convert the parameters theta to hazard ratios in a standard form
# Rows of the outputted matrices correspond to ages 0:80 and columns correspond
# (in order) to non-carriers of the rare genetic variant, heterozygote carriers
```

```
# and homozygote carriers.  We assume below that the variant acts dominantly,
# i.e. that the hazard ratios for heterozygote and homozygote carriers are the same.
calc_HR <- function(theta, sex) {
  # Endometrial cancer -- no age dependence
  age.max <- 80   # The maximum age in the population incidences
  if (sex=="F") {
    hr <- exp(as.numeric(theta["end.f"]))
  } else {
    hr <- 1  # Male population incidences are all 0, so this HR is of no consequence
  }
  HR.end <- cbind(rep(1, age.max+1), rep(hr, age.max+1), rep(hr, age.max+1)) # Dominant
  # Colorectal cancer -- age and sex dependent HRs, piece-wise linear functions of age
  if (sex=="F") {
    hr40 <- exp(theta["col.f.40"])
    hr60 <- exp(theta["col.f.60"])
  }
  if (sex=="M") {
    hr40 <- exp(theta["col.m.40"])
    hr60 <- exp(theta["col.m.60"])
  }
  hr <- c(rep(hr40, 41),  hr40 + (1:20)*(hr60-hr40)/20,  rep(hr60, age.max-60))
  HR.col <- cbind(rep(1, age.max+1), hr, hr)  # Dominant variant

  # Output
  list(col=unname(HR.col), end=unname(HR.end))

}
calc_HR(theta.start, "F")

# Create a dataset to adjust for the clinic-based ascertainment of some families
# (those with dat$clinic == 1) and the population-based ascertainment of others
# (those with dat$clinic == 0).  In both cases, a person (the proband of the
# family, identified by dat$proband == 1) was recruited into the study, then
# their families were subsequently recruited.  Clinic-based probands were recruited
# via family cancer clinics, so they usually have a strong family history of
# cancer, while population-based probands were recruited irrespective of family
# history.  Many probands were tested for the variant of interest, but only
# carrier probands and their families are included in the analysis here.
# For clinic-based families, we condition on the proband's genotype and the
# phenotypes of the entire family, while for population-based families we
# only condition on the proband's genotype and phenotypes.
asc <- dat
asc <- asc[asc$clinic == 1 | asc$proband == 1, ]
asc$geno[asc$clinic == 1 & asc$proband == 0] <- NA  # Clinic-based
asc$mother[asc$clinic == 0] <- NA  # Population-based
asc$father[asc$clinic == 0] <- NA  # Population-based

# Compatible genotypes
# Genotypes of the underlying genetic model correspond 1-to-1 to poss.genos, below
unique(dat$geno)    # In dat$geno, "+" is the genetic variant, "-" is the wildtype
poss.genos <- c("-/-", "+/-", "+/+")    # The possible (non-missing) values of dat$geno
cgenos <- c()
for (i in 1:length(poss.genos))  cgenos <- c(cgenos, list(i))
```

```
names(cgenos) <- poss.genos
cgenos

# Identical twins
# nbm007 and nbm008 are a pair of identical twins, and so are rzp004 and rzp009
mzyg <- list(c("nbm007", "nbm008"), c("rzp004", "rzp009"))

# The number of cores
parallel::detectCores()  # The number of cores available on your computer

# Perform the analysis
## Not run:
  fit_seg <- segregation_analysis(theta.start, dat, geno_freq, trans, calc_inc, calc_HR,
                                  ncores=1, monozyg=mzyg, compat_genos=cgenos,
                                  asc=asc, verbose=TRUE, centre=TRUE)
## End(Not run)
fit_seg <- clipp::fit_seg  # Analysis takes 30s so load previously computed output
#' stats4::summary(fit_seg)
stats4::coef(fit_seg)
exp(fit_seg@coef)
sqrt(diag(stats4::vcov(fit_seg)))
2 * fit_seg@min
```

`trans_monogenic` *The transmission matrix for a single genetic locus*

Description

A function to calculate the transmission matrix for a single autosomal genetic locus with an arbitrary number of alleles and unphased genotypes, based on Mendel's laws of inheritance.

Usage

```
trans_monogenic(n_alleles, annotate = FALSE)
```

Arguments

`n_alleles`	A positive integer, interpreted as the number of possible alleles at the genetic locus.
`annotate`	A logical flag. When `FALSE` (the default), the function returns a matrix suitable to be used as the `trans` argument of `pedigree_loglikelihood`. When `TRUE`, the function annotates this matrix (and converts it to a data frame) to make the output more easily understood by humans.

Details

When annotate is FALSE, this function returns a matrix of genetic transmission probabilities, whose rows corresponding to the possible joint parental genotypes and whose columns corresponding to the possible offspring genotypes. There are ngeno = n_alleles * (n_alleles + 1) / 2 possible unphased genotypes, and by choosing an order on these genotypes (which can be viewed by setting annotate to TRUE, see below) we can label the set of possible genotypes as 1:ngeno. Then the (ngeno * gm + gf - ngeno, go)th element of the outputted matrix is the conditional probability that a person has genotype go, given that his or her biological mother and father have genotypes gm and gf, respectively.

When annotate is TRUE, the function converts this matrix to a data frame, adds column names giving the offspring genotype corresponding to each column, and adds columns gm and gf describing the parental genotypes corresponding to each row. In this data frame, genotypes are written in the usual form 1/1, 1/2, ... for the alleles 1:n_alleles.

Note that if the output of this function is to be used as the trans argument of pedigree_loglikelihood then the annotate option must be set to FALSE.

Value

Either a matrix of genetic transmission probabilities suitable to be used as the trans argument of pedigree_loglikelihood (if annotate is FALSE), or a data frame that is an annotated version of this matrix (if annotate is TRUE).

Examples

```
# The transition matrix for a biallelic, autosomal locus with unphased genotypes
trans_monogenic(2)
trans_monogenic(2, annotate = TRUE)
```

---

trans_phased *The transmission matrix for phased genotypes*

Description

A function to calculate the transmission matrix for a single autosomal genetic locus with an arbitrary number of alleles and phased genotypes, based on Mendel's laws of inheritance. Phased genotypes can be used to investigate parent-of-origin effects, e.g. see (van Vliet et al., 2011).

Usage

```
trans_phased(n_alleles, annotate = FALSE)
```

Arguments

n_alleles	A positive integer, interpreted as the number of possible alleles at the genetic locus.
annotate	A logical flag. When FALSE (the default), the function returns a matrix suitable to be used as the trans argument of pedigree_loglikelihood. When TRUE, the function annotates this matrix (and converts it to a data frame) to make the output more easily understood by humans.

Details

When annotate is FALSE, a matrix of genetic transmission probabilities is returned, with rows corresponding to the possible joint parental genotypes and columns corresponding to the possible offspring genotypes. There are ngeno = n_alleles^2 possible phased genotypes, and by choosing an order on these genotypes (which can be viewed by setting annotate to TRUE, see below) we can label the set of possible phased genotypes as 1:ngeno. Then the (ngeno * gm + gf - ngeno, go)th element of the outputted matrix is the conditional probability that a person has genotype go, given that his or her biological mother and father have genotypes gm and gf, respectively.

When annotate is TRUE, the function converts this matrix to a data frame, adds column names giving the offspring genotype corresponding to each column, and adds columns gm and gf describing the parental genotypes corresponding to each row. In this data frame, phased genotypes are written in the usual form 1|1, 1|2, ... for the alleles 1:n_alleles, where a|b means the maternal allele is a and the paternal allele is b.

Note that if the output of this function is to be used as the trans argument of pedigree_loglikelihood then the annotate option must be set to FALSE.

Value

Either a matrix of genetic transmission probabilities suitable to be used as the trans argument of pedigree_loglikelihood (if annotate is FALSE), or a data frame that is an annotated version of this matrix (if annotate is TRUE).

References

van Vliet CM, Dowty JG, van Vliet JL, et al. Dependence of colorectal cancer risk on the parent-of-origin of mutations in DNA mismatch repair genes. Hum Mutat. 2011;32(2):207-212.

Examples

```
# The transition matrix for a biallelic, autosomal locus with phased genotypes
trans_phased(2)
trans_phased(2, annotate = TRUE)
```

trans_polygenic *The transmission matrix for the hypergeometric polygenic model*

Description

A function to calculate the transmission matrix for the hypergeometric polygenic model of (Cannings et al., 1978), see also Section 8.9 of (Lange, 2002) for a nice description of this model.

Usage

```
trans_polygenic(n_loci, annotate = FALSE)
```

Arguments

n_loci	A positive integer, interpreted as the number of biallelic genetic loci that contribute to the polygene. The polygene will have 2*n_loci + 1 genotypes, so n_loci is typically fairly small, e.g. 4.
annotate	A logical flag. When FALSE (the default), the function returns a matrix suitable to be used as the trans argument of pedigree_loglikelihood. When TRUE, the function annotates this matrix (and converts it to a data frame) to make the output more easily understood by humans.

Details

This function calculates the genetic transmission probabilities (i.e. the conditional probability of a person's genotype, given his or her biological parents' genotypes) for the hypergeometric polygenic model, which is described in geno_freq_polygenic.

When annotate is FALSE, a matrix of transmission probabilities is returned, with rows corresponding to the possible joint parental genotypes and columns corresponding to the possible offspring genotypes. Setting annotate to TRUE shows which rows and columns correspond to which genotypes, by adding offspring genotypes as column names and adding columns gm and gf containing (respectively) the mother's and father's genotypes. Note that if the output of this function is to be used as the trans argument of pedigree_loglikelihood then the annotate option must be set to FALSE.

Value

Either a matrix of genetic transmission probabilities suitable to be used as the trans argument of pedigree_loglikelihood (if annotate is FALSE), or a data frame that is an annotated version of this matrix (if annotate is TRUE).

References

Cannings C, Thompson E, Skolnick M. Probability functions on complex pedigrees. Advances in Applied Probability, 1978;10(1):26-61.

Lange K. Mathematical and Statistical Methods for Genetic Analysis (second edition). Springer, New York. 2002.

Examples

```
trans_polygenic(4, annotate = TRUE)
apply(trans_polygenic(4), 1, sum)
```

Index

∗ **datasets**
    dat_large, 5
    dat_seg, 6
    dat_small, 7
    fit_seg, 7
    inc_seg, 14
    penet_large, 18
    penet_small, 19

clipp *(*clipp-package*)*, 2
clipp-package, 2
combine_loci, 3

dat_large, 5
dat_seg, 6, *26*
dat_small, 7

fit_seg, 7

geno_freq_monogenic, 8, *12*, *15*
geno_freq_phased, 9, *12*, *15*
geno_freq_polygenic, 10, *33*
genotype_probabilities, *2, 3*, 12

inc_seg, 14

pedigree_loglikelihood, *2–5*, *8–11*, *13*, 15, *25–27*, *30–33*
penet_large, 18
penet_small, 19
plot_pedigree, 19, *22*
plot_pedigree_adv, *19*, 22

segregation_analysis, *6, 7*, 25
stats4::mle, *7*, *28*

trans_monogenic, *4*, *12*, *16*, 30
trans_phased, *12*, *16*, 31
trans_polygenic, 33