Package 'LTFHPlus'

November 17, 2025

Type Package

Title Implementation of LT-FH++

Version 2.2.0

Description Implementation of LT-FH++, an extension of
the liability threshold family history (LT-FH) model.
LT-FH++ uses a Gibbs sampler for sampling from the truncated
multivariate normal distribution and allows for
flexible family structures. LT-FH++ was first described in
Pedersen, Emil M., et al. (2022)
<doi:10.1016/j.ajhg.2022.01.009>
as an extension to LT-FH with more flexible family structures,
and again as the age-dependent liability threshold (ADuLT) model
Pedersen, Emil M., et al. (2023)
<https://www.nature.com/articles/s41467-023-41210-z>
as an alternative to traditional time-to-event genome-wide
association studies, where family history was not considered.

License GPL-3

Imports batchmeans, dplyr, future.apply, future, purrr, Rcpp, rlang,
stats, stringr, tibble, tmvtnorm, tidyselect, igraph, xgboost,
tidyr, ggplot2

Suggests MASS, knitr, rmarkdown, kinship2, testthat

LinkingTo Rcpp

Encoding UTF-8

RoxygenNote 7.3.3

Language en-GB

URL https://emilmip.github.io/LTFHPlus/,
https://github.com/EmilMiP/LTFHPlus

BugReports https://github.com/EmilMiP/LTFHPlus/issues

NeedsCompilation yes

Author Emil Michael Pedersen [aut, cre],
Florian Privé [aut, ths],
Bjarni Jóhann Vilhjálmsson [ths],
Esben Agerbo [ths],
Jette Steinbach [aut],
Lucas Rasmussen [ctb]

Maintainer Emil Michael Pedersen <emp@ph.au.dk>

Repository CRAN

Date/Publication 2025-11-17 10:10:02 UTC

Contents

`attach_attributes` *Attach attributes to a family graphs*

Description

This function attaches attributes to family graphs, such as lower and upper thresholds, for each family member. This allows for a user-friendly way to attach personalised thresholds and other per-family specific attributes to the family graphs.

Usage

```
attach_attributes(
  cur_fam_graph,
  cur_proband,
  pid,
  attr_tbl,
  attr_names,
  proband_cols_to_censor = NA
)
```

Arguments

`cur_fam_graph`	An igraph object (neighbourhood graph around a proband) with family members up to degree n.
`cur_proband`	Current proband id (center of the neighbourhood graph).
`pid`	Column name of personal id (within a family).
`attr_tbl`	Tibble with family id and attributes for each family member.
`attr_names`	Names of attributes to be assigned to each node (family member) in the graph.
`proband_cols_to_censor`	Which columns should be made uninformative for the proband? Defaults to NA. Used to exclude proband's information for prediction with, e.g. c("lower", "upper").

Value

igraph object (neighbourhood graph around a proband) with updated attributes for each node in the graph.

construct_covmat *Constructing a covariance matrix for a variable number of phenotypes*

Description

construct_covmat returns the covariance matrix for an underlying target individual and a variable number of its family members for a variable number of phenotypes. It is a wrapper around construct_covmat_single and construct_covmat_multi.

Usage

```
construct_covmat(
  fam_vec = c("m", "f", "s1", "mgm", "mgf", "pgm", "pgf"),
  n_fam = NULL,
  add_ind = TRUE,
  h2 = 0.5,
  genetic_corrmat = NULL,
  full_corrmat = NULL,
  phen_names = NULL
)
```

Arguments

fam_vec	A vector of strings holding the different family members. All family members must be represented by strings from the following list: • m (Mother) • f (Father) • c[0-9]*.[0-9]* (Children) • mgm (Maternal grandmother) • mgf (Maternal grandfather) • pgm (Paternal grandmother) • pgf (Paternal grandfather) • s[0-9]* (Full siblings) • mhs[0-9]* (Half-siblings - maternal side) • phs[0-9]* (Half-siblings - paternal side) • mau[0-9]* (Aunts/Uncles - maternal side) • pau[0-9]* (Aunts/Uncles - paternal side). Defaults to c("m","f","s1","mgm","mgf","pgm","pgf").
n_fam	A named vector holding the desired number of family members. See setNames. All names must be picked from the list mentioned above. Defaults to NULL.
add_ind	A logical scalar indicating whether the genetic component of the full liability as well as the full liability for the underlying individual should be included in the covariance matrix. Defaults to TRUE.

h2 Either a number representing the heritability on liability scale for one single phenotype or a numeric vector representing the liability-scale heritabilities for a positive number of phenotypes. All entries in h2 must be non-negative and at most 1.

genetic_corrmat
Either NULL or a numeric matrix holding the genetic correlations between the desired phenotypes. All diagonal entries must be equal to one, while all off-diagonal entries must be between -1 and 1. In addition, the matrix must be symmetric. Defaults to NULL.

full_corrmat Either NULL or a numeric matrix holding the full correlations between the desired phenotypes. All diagonal entries must be equal to one, while all off-diagonal entries must be between -1 and 1. In addition, the matrix must be symmetric. Defaults to NULL.

phen_names Either NULL or a character vector holding the phenotype names. These names will be used to create the row and column names for the covariance matrix. If it is not specified, the names will default to phenotype1, phenotype2, etc. Defaults to NULL.

Details

This function can be used to construct a covariance matrix for a given number of family members. If h2 is a number, each entry in this covariance matrix equals the percentage of shared DNA between the corresponding individuals times the liability-scale heritability

$$h^2$$

. However, if h2 is a numeric vector, and genetic_corrmat and full_corrmat are two symmetric correlation matrices, each entry equals either the percentage of shared DNA between the corresponding individuals times the liability-scale heritability

$$h^2$$

or the percentage of shared DNA between the corresponding individuals times the correlation between the corresponding phenotypes. The family members can be specified using one of two possible formats.

Value

If either fam_vec or n_fam is used as the argument, if it is of the required format, if add_ind is a logical scalar and h2 is a number satisfying

$$0 \leq h2 \leq 1$$

, then the function construct_covmat will return a named covariance matrix, which row- and column-number corresponds to the length of fam_vec or n_fam (+ 2 if add_ind=TRUE). However, if h2 is a numeric vector satisfying

$$0 \leq h2_i \leq 1$$

for all

$$i \in \{1, ..., n_pheno\}$$

and if genetic_corrmat and full_corrmat are two numeric and symmetric matrices satisfying that all diagonal entries are one and that all off-diagonal entries are between -1 and 1, then construct_covmat will return a named covariance matrix, which number of rows and columns corresponds to the number of phenotypes times the length of fam_vec or n_fam (+ 2 if add_ind=TRUE). If both fam_vec and n_fam are equal to c() or NULL, the function returns either a 2×2 matrix holding only the correlation between the genetic component of the full liability and the full liability for the individual under consideration, or a

$$(2 \times n_pheno) \times (2 \times n_pheno)$$

matrix holding the correlation between the genetic component of the full liability and the full liability for the underlying individual for all phenotypes. If both fam_vec and n_fam are specified, the user is asked to decide on which of the two vectors to use. Note that the returned object has different attributes, such as fam_vec, n_fam, add_ind and h2.

See Also

get_relatedness, construct_covmat_single, construct_covmat_multi

Examples

```
construct_covmat()
construct_covmat(fam_vec = c("m","mgm","mgf","mhs1","mhs2","mau1"),
                 n_fam = NULL,
                 add_ind = TRUE,
                 h2 = 0.5)
construct_covmat(fam_vec = NULL,
                 n_fam = stats::setNames(c(1,1,1,2,2), c("m","mgm","mgf","s","mhs")),
                 add_ind = FALSE,
                 h2 = 0.3)
construct_covmat(h2 = c(0.5,0.5), genetic_corrmat = matrix(c(1,0.4,0.4,1), nrow = 2),
                 full_corrmat = matrix(c(1,0.6,0.6,1), nrow = 2))
```

---

construct_covmat_multi

Constructing a covariance matrix for multiple phenotypes

Description

construct_covmat_multi returns the covariance matrix for an underlying target individual and a variable number of its family members for multiple phenotypes.

Usage

```
construct_covmat_multi(
  fam_vec = c("m", "f", "s1", "mgm", "mgf", "pgm", "pgf"),
  n_fam = NULL,
  add_ind = TRUE,
```

```
  genetic_corrmat,
  full_corrmat,
  h2_vec,
  phen_names = NULL
)
```

Arguments

fam_vec	A vector of strings holding the different family members. All family members must be represented by strings from the following list: • m (Mother) • f (Father) • c[0-9]*.[0-9]* (Children) • mgm (Maternal grandmother) • mgf (Maternal grandfather) • pgm (Paternal grandmother) • pgf (Paternal grandfather) • s[0-9]* (Full siblings) • mhs[0-9]* (Half-siblings - maternal side) • phs[0-9]* (Half-siblings - paternal side) • mau[0-9]* (Aunts/Uncles - maternal side) • pau[0-9]* (Aunts/Uncles - paternal side). Defaults to c("m","f","s1","mgm","mgf","pgm","pgf").
n_fam	A named vector holding the desired number of family members. See setNames. All names must be picked from the list mentioned above. Defaults to NULL.
add_ind	A logical scalar indicating whether the genetic component of the full liability as well as the full liability for the underlying individual should be included in the covariance matrix. Defaults to TRUE.
genetic_corrmat	A numeric matrix holding the genetic correlations between the desired phenotypes. All diagonal entries must be equal to one, while all off-diagonal entries must be between -1 and 1. In addition, the matrix must be symmetric.
full_corrmat	A numeric matrix holding the full correlations between the desired phenotypes. All diagonal entries must be equal to one, while all off-diagonal entries must be between -1 and 1. In addition, the matrix must be symmetric.
h2_vec	A numeric vector representing the liability-scale heritabilities for all phenotypes. All entries in h2_vec must be non-negative and at most 1.
phen_names	A character vector holding the phenotype names. These names will be used to create the row and column names for the covariance matrix. If it is not specified, the names will default to phenotype1, phenotype2, etc. Defaults to NULL.

Details

This function can be used to construct a covariance matrix for a given number of family members. Each entry in this covariance matrix equals either the percentage of shared DNA between the corresponding individuals times the liability-scale heritability h^2 or the percentage of shared DNA

between the corresponding individuals times the correlation between the corresponding phenotypes. That is, for the same phenotype, the covariance between all combinations of the genetic component of the full liability and the full liability is given by

$$\text{Cov}\left(l_g, l_g\right) = h^2,$$

$$\text{Cov}\left(l_g, l_o\right) = h^2,$$

$$\text{Cov}\left(l_o, l_g\right) = h^2$$

and

$$\text{Cov}\left(l_o, l_o\right) = 1.$$

For two different phenotypes, the covariance is given by

$$\text{Cov}\left(l_g^1, l_g^2\right) = \rho_g^{1,2},$$

$$\text{Cov}\left(l_g^1, l_o^2\right) = \rho_g^{1,2},$$

$$\text{Cov}\left(l_o^1, l_g^2\right) = \rho_g^{1,2}$$

and

$$\text{Cov}\left(l_o^1, l_o^2\right) = \rho_g^{1,2} + \rho_e^{1,2},$$

where l_g^i and l_o^i are the genetic component of the full liability and the full liability for phenotype i, respectively, $\rho_g^{i,j}$ is the genetic correlation between phenotype i and j and $\rho_e^{1,2}$ is the environmental correlation between phenotype i and j. The family members can be specified using one of two possible formats.

Value

If either `fam_vec` or `n_fam` is used as the argument and if it is of the required format, if `genetic_corrmat` and `full_corrmat` are two numeric and symmetric matrices satisfying that all diagonal entries are one and that all off-diagonal entries are between -1 and 1, and if `h2_vec` is a numeric vector satisfying $0 \leq h2_i \leq 1$ for all $i \in \{1, ..., n_pheno\}$, then the output will be a named covariance matrix. The number of rows and columns corresponds to the number of phenotypes times the length of `fam_vec` or `n_fam` (+ 2 if `add_ind=TRUE`). If both `fam_vec` and `n_fam` are equal to `c()` or `NULL`, the function returns a $(2 \times n_pheno) \times (2 \times n_pheno)$ matrix holding only the correlation between the genetic component of the full liability and the full liability for the underlying individual for all phenotypes. If both `fam_vec` and `n_fam` are specified, the user is asked to decide on which of the two vectors to use. Note that the returned object has a number different attributes,namely `fam_vec`, `n_fam`, `add_ind`, `genetic_corrmat`, `full_corrmat`, `h2` and `phenotype_names`.

See Also

`get_relatedness`, `construct_covmat_single` and `construct_covmat`.

Examples

```
construct_covmat_multi(fam_vec = NULL,
                       genetic_corrmat = matrix(c(1, 0.5, 0.5, 1), nrow = 2),
                       full_corrmat = matrix(c(1, 0.55, 0.55, 1), nrow = 2),
                       h2_vec = c(0.37,0.44),
                       phen_names = c("p1","p2"))
construct_covmat_multi(fam_vec = c("m","mgm","mgf","mhs1","mhs2","mau1"),
                       n_fam = NULL,
                       add_ind = TRUE,
                       genetic_corrmat = diag(3),
                       full_corrmat = diag(3),
                       h2_vec = c(0.8, 0.65))
construct_covmat_multi(fam_vec = NULL,
                     n_fam = stats::setNames(c(1,1,1,2,2), c("m","mgm","mgf","s","mhs")),
                       add_ind = FALSE,
                       genetic_corrmat = diag(2),
                       full_corrmat = diag(2),
                       h2_vec = c(0.75,0.85))
```

---

construct_covmat_single

Constructing a covariance matrix for a single phenotype

Description

`construct_covmatc_single` returns the covariance matrix for an underlying target individual and a variable number of its family members

Usage

```
construct_covmat_single(
  fam_vec = c("m", "f", "s1", "mgm", "mgf", "pgm", "pgf"),
  n_fam = NULL,
  add_ind = TRUE,
  h2 = 0.5
)
```

Arguments

`fam_vec` A vector of strings holding the different family members. All family members must be represented by strings from the following list:

- `m` (Mother)
- `f` (Father)
- `c[0-9]*.[0-9]*` (Children)
- `mgm` (Maternal grandmother)
- `mgf` (Maternal grandfather)

- pgm (Paternal grandmother)
- pgf (Paternal grandfather)
- s[0-9]* (Full siblings)
- mhs[0-9]* (Half-siblings - maternal side)
- phs[0-9]* (Half-siblings - paternal side)
- mau[0-9]* (Aunts/Uncles - maternal side)
- pau[0-9]* (Aunts/Uncles - paternal side).

`n_fam` A named vector holding the desired number of family members. See `setNames`. All names must be picked from the list mentioned above. Defaults to NULL.

`add_ind` A logical scalar indicating whether the genetic component of the full liability as well as the full liability for the underlying individual should be included in the covariance matrix. Defaults to TRUE.

`h2` A number representing the squared heritability on liability scale for a single phenotype. Must be non-negative and at most 1. Defaults to 0.5.

Details

This function can be used to construct a covariance matrix for a given number of family members. Each entry in this covariance matrix equals the percentage of shared DNA between the corresponding individuals times the liability-scale heritability h^2. The family members can be specified using one of two possible formats.

Value

If either `fam_vec` or `n_fam` is used as the argument, if it is of the required format and `h2` is a number satisfying $0 \leq h2 \leq 1$, then the output will be a named covariance matrix. The number of rows and columns corresponds to the length of `fam_vec` or `n_fam` (+ 2 if `add_ind=TRUE`). If both `fam_vec = c()/NULL` and `n_fam = c()/NULL`, the function returns a 2×2 matrix holding only the correlation between the genetic component of the full liability and the full liability for the individual. If both `fam_vec` and `n_fam` are given, the user is asked to decide on which of the two vectors to use. Note that the returned object has different attributes, such as `fam_vec`, `n_fam`, `add_ind` and `h2`.

See Also

`get_relatedness`, `construct_covmat_multi`, `construct_covmat`

Examples

```
construct_covmat_single()
construct_covmat_single(fam_vec = c("m","mgm","mgf","mhs1","mhs2","mau1"),
n_fam = NULL, add_ind = TRUE, h2 = 0.5)
construct_covmat_single(fam_vec = NULL, n_fam = stats::setNames(c(1,1,1,2,2),
c("m","mgm","mgf","s","mhs")), add_ind = FALSE, h2 = 0.3)
```

convert_age_to_cir *Convert age to cumulative incidence rate*

Description

convert_age_to_cir computes the cumulative incidence rate from a person's age.

Usage

```
convert_age_to_cir(age, pop_prev = 0.1, mid_point = 60, slope = 1/8)
```

Arguments

age	A non-negative number representing the individual's age.
pop_prev	A positive number representing the overall population prevalence. Must be at most 1. Defaults to 0.1.
mid_point	A positive number representing the mid point logistic function. Defaults to 60.
slope	A number holding the rate of increase. Defaults to 1/8.

Details

Given a person's age, convert_age_to_cir can be used to compute the cumulative incidence rate (cir), which is given by the formula

$$pop\_prev/(1 + exp((mid\_point - age) * slope))$$

Value

If age and mid_point are positive numbers, if pop_prev is a positive number between 0 and 1 and if slope is a valid number, then convert_age_to_cir returns a number, which is equal to the cumulative incidence rate.

Examples

```
curve(sapply(age, convert_age_to_cir), from = 10, to = 110, xname = "age")
```

`convert_age_to_thresh` *Convert age to threshold*

Description

`convert_age_to_thresh` computes the threshold from a person's age using either the logistic function or the truncated normal distribution

Usage

```
convert_age_to_thresh(
  age,
  dist = "logistic",
  pop_prev = 0.1,
  mid_point = 60,
  slope = 1/8,
  min_age = 10,
  max_age = 90,
  lower = stats::qnorm(0.05, lower.tail = FALSE),
  upper = Inf
)
```

Arguments

`age`	A non-negative number representing the individual's age.
`dist`	A string indicating which distribution to use. If dist = "logistic", the logistic function will be used to compute the age of onset. If dist = "normal", the truncated normal distribution will be used instead. Defaults to "logistic".
`pop_prev`	Only necessary if dist = "logistic". A positive number representing the overall population prevalence. Must be at most 1. Defaults to 0.1.
`mid_point`	Only necessary if dist = "logistic". A positive number representing the mid point logistic function. Defaults to 60.
`slope`	Only necessary if dist = "logistic". A number holding the rate of increase. Defaults to 1/8.
`min_age`	Only necessary if dist = "normal". A positive number representing the individual's earliest age. Defaults to 10.
`max_age`	Only necessary if dist = "normal". A positive number representing the individual's latest age. Must be greater than min_aoo. Defaults to 90.
`lower`	Only necessary if dist = "normal". A number representing the lower cutoff point for the truncated normal distribution. Defaults to 1.645 (stats::qnorm(0.05, lower.tail = FALSE)).
`upper`	Only necessary if dist = "normal". A number representing the upper cutoff point of the truncated normal distribution. Must be greater or equal to lower. Defaults to Inf.

Details

Given a person's age, `convert_age_to_thresh` can be used to first compute the cumulative incidence rate (cir), which is then used to compute the threshold using either the logistic function or the truncated normal distribution. Under the logistic function, the formula used to compute the threshold from an individual's age is given by

$$qnorm(pop\_prev/(1 + exp((mid\_point - age) * slope)), lower.tail = F)$$

, while it is given by

$$qnorm((1-(age-min\_age)/max\_age)*(pnorm(upper)-pnorm(lower))+pnorm(lower))$$

under the truncated normal distribution.

Value

If age is a positive number and all other necessary arguments are valid, then `convert_age_to_thresh` returns a number, which is equal to the threshold.

Examples

```
curve(sapply(age, convert_age_to_thresh), from = 10, to = 110, xname = "age")
```

---

convert_cir_to_age *Convert cumulative incidence rate to age*

Description

`convert_cir_to_age` computes the age from a person's cumulative incidence rate.

Usage

```
convert_cir_to_age(cir, pop_prev = 0.1, mid_point = 60, slope = 1/8)
```

Arguments

`cir`	A positive number representing the individual's cumulative incidence rate.
`pop_prev`	A positive number representing the overall population prevalence. Must be at most 1 and must be larger than cir. Defaults to 0.1.
`mid_point`	A positive number representing the mid point logistic function. Defaults to 60.
`slope`	A number holding the rate of increase. Defaults to 1/8.

Details

Given a person's cumulative incidence rate (cir), `convert_cir_to_age` can be used to compute the corresponding age, which is given by

$$mid\_point - \log(pop\_prev/cir - 1) * 1/slope$$

Value

If cir and mid_point are positive numbers, if pop_prev is a positive number between 0 and 1 and if slope is a valid number, then `convert_cir_to_age` returns a number, which is equal to the current age.

Examples

```
curve(sapply(cir, convert_cir_to_age), from = 0.001, to = 0.099, xname = "cir")
```

---

convert_format *Attempts to convert the list entry input format to a long format*

Description

Attempts to convert the list entry input format to a long format

Usage

```
convert_format(family, threshs, personal_id_col = "pid", role_col = NULL)
```

Arguments

`family`	a tibble with two entries, family id and personal id. personal id should end in "_role", if a role column is not present.
`threshs`	thresholds, with a personal id (without role) as well as the lower and upper thresholds
`personal_id_col`	column name that holds the personal id
`role_col`	column name that holds the role

Value

returns a format similar to `prepare_LTFHPlus_input`, which is used by `estimate_liability`

Examples

```
family <- data.frame(
fam_id = c(1, 1, 1, 1),
pid = c(1, 2, 3, 4),
role = c("o", "m", "f", "pgf")
)

threshs <- data.frame(
  pid = c(1, 2, 3, 4),
  lower = c(-Inf, -Inf, 0.8, 0.7),
  upper = c(0.8, 0.8, 0.8, 0.7)
)
```

```
convert_format(family, threshs)
```

`convert_liability_to_aoo` *Convert liability to age of onset*

Description

`convert_liability_to_aoo` computes the age of onset from an individual's true underlying liability using either the logistic function or the truncated normal distribution.

Usage

```
convert_liability_to_aoo(
  liability,
  dist = "logistic",
  pop_prev = 0.1,
  mid_point = 60,
  slope = 1/8,
  min_aoo = 10,
  max_aoo = 90,
  lower = stats::qnorm(0.05, lower.tail = FALSE),
  upper = Inf
)
```

Arguments

`liability`	A number representing the individual's true underlying liability.
`dist`	A string indicating which distribution to use. If dist = "logistic", the logistic function will be used to compute the age of onset. If dist = "normal", the truncated normal distribution will be used instead. Defaults to "logistic".
`pop_prev`	Only necessary if dist = "logistic". A positive number representing the overall population prevalence. Must be at most 1. Defaults to 0.1.
`mid_point`	Only necessary if dist = "logistic". A positive number representing the mid point logistic function. Defaults to 60.
`slope`	Only necessary if dist = "logistic". A number holding the rate of increase. Defaults to 1/8.
`min_aoo`	Only necessary if dist = "normal". A positive number representing the individual's earliest age of onset. Defaults to 10.
`max_aoo`	Only necessary if dist = "normal". A positive number representing the individual's latest age of onset. Must be greater than min_aoo. Defaults to 90.
`lower`	Only necessary if dist = "normal". A number representing the lower cutoff point for the truncated normal distribution. Defaults to 1.645 (stats::qnorm(0.05, lower.tail = FALSE)).

upper — Only necessary if dist = "normal". A number representing the upper cutoff point of the truncated normal distribution. Must be greater or equal to lower. Defaults to Inf.

Details

Given a person's cumulative incidence rate (cir), convert_liability_to_aoo can be used to compute the corresponding age. Under the logistic function, the age is given by

$$mid\_point - log(pop\_prev/cir - 1) * 1/slope$$

, while it is given by

$$(1 - truncated\_normal\_cdf(liability = liability, lower = lower, upper = upper)) * max\_aoo + min\_aoo$$

under the truncated normal distribution.

Value

If liability is a number and all other necessary arguments are valid, then convert_liability_to_aoo returns a positive number, which is equal to the age of onset.

Examples

```
curve(sapply(liability, convert_liability_to_aoo), from = 1.3, to = 3.5, xname = "liability")
curve(sapply(liability, convert_liability_to_aoo, dist = "normal"),
 from = qnorm(0.05, lower.tail = FALSE), to = 3.5, xname = "liability")
```

---

convert_observed_to_liability_scale

Convert the heritability on the observed scale to that on the liability scale

---

Description

convert_observed_to_liability_scale transforms the heritability on the observed scale to the heritability on the liability scale.

Usage

```
convert_observed_to_liability_scale(
  obs_h2 = 0.5,
  pop_prev = 0.05,
  prop_cases = 0.5
)
```

Arguments

obs_h2	A number or numeric vector representing the liability-scale heritability(ies)on the observed scale. Must be non-negative and at most 1. Defaults to 0.5
pop_prev	A number or numeric vector representing the population prevalence(s). All entries must be non-negative and at most one. If it is a vector, it must have the same length as obs_h2. Defaults to 0.05.
prop_cases	Either NULL or a number or a numeric vector representing the proportion of cases in the sample. All entries must be non-negative and at most one. If it is a vector, it must have the same length as obs_h2. Defaults to 0.5.

Details

This function can be used to transform the heritability on the observed scale to that on the liability scale. convert_observed_to_liability_scale uses either Equation 17 (if prop_cases = NULL) or Equation 23 from Sang Hong Lee, Naomi R. Wray, Michael E. Goddard and Peter M. Visscher, "Estimating Missing Heritability for Diseases from Genome-wide Association Studies", The American Journal of Human Genetics, Volume 88, Issue 3, 2011, pp. 294-305, doi:10.1016/j.ajhg.2011.02.002 to transform the heritability on the observed scale to the heritability on the liability scale.

Value

If obs_h2, pop_prev and prop_cases are non-negative numbers that are at most one, the function returns the heritability on the liability scale using Equation 23 from Sang Hong Lee, Naomi R. Wray, Michael E. Goddard and Peter M. Visscher, "Estimating Missing Heritability for Diseases from Genome-wide Association Studies", The American Journal of Human Genetics, Volume 88, Issue 3, 2011, pp. 294-305, doi:10.1016/j.ajhg.2011.02.002. If obs_h2, pop_prev and prop_cases are non-negative numeric vectors where all entries are at most one, the function returns a vector of the same length as obs_h2. Each entry holds to the heritability on the liability scale which was obtained from the corresponding entry in obs_h2 using Equation 23. If obs_h2 and pop_prev are non-negative numbers that are at most one and prop_cases is NULL, the function returns the heritability on the liability scale using Equation 17 from Sang Hong Lee, Naomi R. Wray, Michael E. Goddard and Peter M. Visscher, "Estimating Missing Heritability for Diseases from Genome-wide Association Studies", The American Journal of Human Genetics, Volume 88, Issue 3, 2011, pp. 294-305, doi:10.1016/j.ajhg.2011.02.002. If obs_h2 and pop_prev are non-negative numeric vectors such that all entries are at most one, while prop_cases is NULL, convert_observed_to_liability_scale returns a vector of the same length as obq_h2. Each entry holds to the liability-scale heritability that was obtained from the corresponding entry in obs_h2 using Equation 17.

References

Sang Hong Lee, Naomi R. Wray, Michael E. Goddard, Peter M. Visscher (2011, March). Estimating Missing Heritability for Diseases from Genome-wide Association Studies. In The American Journal of Human Genetics (Vol. 88, Issue 3, pp. 294-305). doi:10.1016/j.ajhg.2011.02.002

Examples

```
convert_observed_to_liability_scale()
convert_observed_to_liability_scale(prop_cases=NULL)
convert_observed_to_liability_scale(obs_h2 = 0.8, pop_prev = 1/44,
                                    prop_cases = NULL)
convert_observed_to_liability_scale(obs_h2 = c(0.5,0.8),
                                    pop_prev = c(0.05, 1/44),
                                    prop_cases = NULL)
```

---

correct_positive_definite

Positive definite matrices

Description

`correct_positive_definite` verifies that a given covariance matrix is indeed positive definite by checking that all eigenvalues are positive. If the given covariance matrix is not positive definite, `correct_positive_definite` tries to modify the underlying correlation matrices genetic_corrmat and full_corrmat in order to obtain a positive definite covariance matrix.

Usage

```
correct_positive_definite(
  covmat,
  correction_val = 0.99,
  correction_limit = 100
)
```

Arguments

`covmat`	A symmetric and numeric matrix. If the covariance matrix should be corrected, it must have a number of attributes, such as `attr(covmat,"fam_vec")`, `attr(covmat,"n_fam")`, `attr(covmat,"add_ind")`, `attr(covmat,"h2")`, `attr(covmat,"genetic_corrmat")`, `attr(covmat,"full_corrmat")` and `attr(covmat,"phenotype_names")`. Any covariance matrix obtained by `construct_covmat`, `construct_covmat_single` or `construct_covmat_multi` will have these attributes by default.
`correction_val`	A positive number representing the amount by which `genetic_corrmat` and `full_corrmat` will be changed, if some eigenvalues are non-positive. That is, correction_val is the number that will be multiplied to all off_diagonal entries in `genetic_corrmat` and `full_corrmat`. Defaults to 0.99.
`correction_limit`	A positive integer representing the upper limit for the correction procedure. Defaults to 100.

Details

This function can be used to verify that a given covariance matrix is positive definite. It calculates all eigenvalues in order to investigate whether they are all positive. This property is necessary for the covariance matrix to be used as a Gaussian covariance matrix. It is especially useful to check whether any covariance matrix obtained by `construct_covmat_multi` is positive definite. If the given covariance matrix is not positive definite, `correct_positive_definite` tries to modify the underlying correlation matrices (called `genetic_corrmat` and `full_corrmat` in `construct_covmat` or `construct_covmat_multi`) by multiplying all off-diagonal entries in the correlation matrices by a given number.

Value

If `covmat` is a symmetric and numeric matrix and all eigenvalues are positive, `correct_positive_definite` simply returns `covmat`. If some eigenvalues are not positive and `correction_val` is a positive number, `correct_positive_definite` tries to convert `covmat` into a positive definite matrix. If `covmat` has attributes `add_ind`, `h2`, `genetic_corrmat`, `full_corrmat` and `phenotype_names`, `correct_positive_definite` computes a new covariance matrix using slightly modified correlation matrices `genetic_corrmat` and `full_corrmat`. If the correction is performed successfully, i.e. if the new covariance matrix is positive definite,the new covariance matrix is returned. Otherwise, `correct_positive_definite` returns the original covariance matrix.

See Also

`construct_covmat`, `construct_covmat_single` and `construct_covmat_multi`.

Examples

```
ntrait <- 2
genetic_corrmat <- matrix(0.6, ncol = ntrait, nrow = ntrait)
diag(genetic_corrmat) <- 1
full_corrmat <- matrix(-0.25, ncol = ntrait, nrow = ntrait)
diag(full_corrmat) <- 1
h2_vec <- rep(0.6, ntrait)
cov <- construct_covmat(fam_vec = c("m", "f"),
  genetic_corrmat = genetic_corrmat,
  h2 = h2_vec,
  full_corrmat = full_corrmat)
cov
correct_positive_definite(cov)
```

---

estimate_gen_liability_ltfh

Estimate genetic liability similar to LT-FH

Description

Estimate genetic liability similar to LT-FH

Usage

```
estimate_gen_liability_ltfh(
  h2,
  phen,
  child_threshold,
  parent_threshold,
  status_col_offspring = "CHILD_STATUS",
  status_col_father = "P1_STATUS",
  status_col_mother = "P2_STATUS",
  status_col_siblings = "SIB_STATUS",
  number_of_siblings_col = "NUM_SIBS",
  tol = 0.01
)
```

Arguments

h2	Liability scale heritability of the trait being analysed.
phen	tibble or data.frame with status of the genotyped individual, parents and siblings.
child_threshold	single numeric value that is used as threshold for the offspring and siblings.
parent_threshold	single numeric value that is used as threshold for both parents
status_col_offspring	Column name of status for the offspring
status_col_father	Column name of status for the father
status_col_mother	Column name of status for the mother
status_col_siblings	Column name of status for the siblings
number_of_siblings_col	Column name for the number of siblings for a given individual
tol	Convergence criteria of the Gibbs sampler. Default is 0.01, meaning a standard error of the mean below 0.01

Value

Returns the estimated genetic liabilities.

Examples

```
phen <- data.frame(
CHILD_STATUS = c(0,0),
P1_STATUS = c(1,1),
P2_STATUS = c(0,1),
SIB_STATUS = c(1,0),
NUM_SIBS = c(2,0))
```

```
h2 <- 0.5
child_threshold <- 0.7
parent_threshold <- 0.8

estimate_gen_liability_ltfh(h2, phen, child_threshold, parent_threshold)
```

`estimate_liability` *Estimating the genetic or full liability for a variable number of phenotypes*

Description

`estimate_liability` estimates the genetic component of the full liability and/or the full liability for a number of individuals based on their family history for one or more phenotypes. It is a wrapper around `estimate_liability_single` and `estimate_liability_multi`.

Usage

```
estimate_liability(
  .tbl = NULL,
  family_graphs = NULL,
  h2 = 0.5,
  pid = "PID",
  fam_id = "fam_ID",
  role = "role",
  family_graphs_col = "fam_graph",
  out = c(1),
  tol = 0.01,
  genetic_corrmat = NULL,
  full_corrmat = NULL,
  phen_names = NULL
)
```

Arguments

`.tbl` A matrix, list or data frame that can be converted into a tibble. Must have at least five columns that hold the family identifier, the personal identifier, the role and the lower and upper thresholds for all phenotypes of interest. Note that the role must be one of the following abbreviations

- `g` (Genetic component of full liability)
- `o` (Full liability)
- `m` (Mother)
- `f` (Father)
- `c[0-9]*.[0-9]*` (Children)
- `mgm` (Maternal grandmother)

- mgf (Maternal grandfather)
- pgm (Paternal grandmother)
- pgf (Paternal grandfather)
- s[0-9]* (Full siblings)
- mhs[0-9]* (Half-siblings - maternal side)
- phs[0-9]* (Half-siblings - paternal side)
- mau[0-9]* (Aunts/Uncles - maternal side)
- pau[0-9]* (Aunts/Uncles - paternal side). Defaults to NULL.

family_graphs A tibble with columns pid and family_graph_col. See prepare_graph for construction of the graphs. The family graphs Defaults to NULL.

h2 Either a number representing the heritability on liability scale for a single phenotype, or a numeric vector representing the liability-scale heritabilities for all phenotypes. All entries in h2 must be non-negative and at most 1.

pid A string holding the name of the column in family and threshs that hold the personal identifier(s). Defaults to "PID".

fam_id A string holding the name of the column in family that holds the family identifier. Defaults to "fam_ID".

role A string holding the name of the column in .tbl that holds the role.Each role must be chosen from the following list of abbreviations

- g (Genetic component of full liability)
- o (Full liability)
- m (Mother)
- f (Father)
- c[0-9]*.[0-9]* (Children)
- mgm (Maternal grandmother)
- mgf (Maternal grandfather)
- pgm (Paternal grandmother)
- pgf (Paternal grandfather)
- s[0-9]* (Full siblings)
- mhs[0-9]* (Half-siblings - maternal side)
- phs[0-9]* (Half-siblings - paternal side)
- mau[0-9]* (Aunts/Uncles - maternal side)
- pau[0-9]* (Aunts/Uncles - paternal side). Defaults to "role".

family_graphs_col

Name of column with family graphs in family_graphs. Defaults to "fam_graph".

out A character or numeric vector indicating whether the genetic component of the full liability, the full liability or both should be returned. If out = c(1) or out = c("genetic"), the genetic liability is estimated and returned. If out = c(2) or out = c("full"), the full liability is estimated and returned. If out = c(1,2) or out = c("genetic", "full"), both components are estimated and returned. Defaults to c(1).

tol A number that is used as the convergence criterion for the Gibbs sampler. Equals the standard error of the mean. That is, a tolerance of 0.2 means that the standard error of the mean is below 0.2. Defaults to 0.01.

genetic_corrmat
: Either NULL (if h2 is a number) or a numeric matrix (if h2 is a vector of length > 1) holding the genetic correlations between the desired phenotypes. All diagonal entries must be equal to one, while all off-diagonal entries must be between -1 and 1. In addition, the matrix must be symmetric. Defaults to NULL.

full_corrmat
: Either NULL (if h2 is a number) or a numeric matrix (if h2 is a vector of length > 1) holding the full correlations between the desired phenotypes. All diagonal entries must be equal to one, while all off-diagonal entries must be between -1 and 1. In addition, the matrix must be symmetric. Defaults to NULL.

phen_names
: Either NULL or a character vector holding the phenotype names. These names will be used to create the row and column names for the covariance matrix. If it is not specified, the names will default to phenotype1, phenotype2, etc. Defaults to NULL.

Details

This function can be used to estimate either the genetic component of the full liability, the full liability or both for a variable number of traits.

Value

If family and threshs are two matrices, lists or data frames that can be converted into tibbles, if family has two columns named like the strings represented in pid and fam_id, if threshs has a column named like the string given in pid as well as a column named "lower" and a column named "upper" and if the liability-scale heritability h2 is a number (length(h2)=1), and out, tol and always_add are of the required form, then the function returns a tibble with either four or six columns (depending on the length of out). The first two columns correspond to the columns fam_id and pid ' present in family. If out is equal to c(1) or c("genetic"), the third and fourth column hold the estimated genetic liability as well as the corresponding standard error, respectively. If out equals c(2) or c("full"), the third and fourth column hold the estimated full liability as well as the corresponding standard error, respectively. If out is equal to c(1,2) or c("genetic","full"), the third and fourth column hold the estimated genetic liability as well as the corresponding standard error, respectively, while the fifth and sixth column hold the estimated full liability as well as the corresponding standard error, respectively. If h2 is a numeric vector of length greater than 1 and if genetic_corrmat, full_corrmat, out and tol are of the required form, then the function returns a tibble with at least six columns (depending on the length of out). The first two columns correspond to the columns fam_id and pid present in the tibble family. If out is equal to c(1) or c("genetic"), the third and fourth columns hold the estimated genetic liability as well as the corresponding standard error for the first phenotype, respectively. If out equals c(2) or c("full"), the third and fourth columns hold the estimated full liability as well as the corresponding standard error for the first phenotype, respectively. If out is equal to c(1,2) or c("genetic","full"), the third and fourth columns hold the estimated genetic liability as well as the corresponding standard error for the first phenotype, respectively, while the fifth and sixth columns hold the estimated full liability as well as the corresponding standard error for the first phenotype, respectively. The remaining columns hold the estimated genetic liabilities and/or the estimated full liabilities as well as the corresponding standard errors for the remaining phenotypes.

See Also

future_apply, estimate_liability_single, estimate_liability_multi

Examples

```
genetic_corrmat <- matrix(0.4, 3, 3)
diag(genetic_corrmat) <- 1
full_corrmat <- matrix(0.6, 3, 3)
diag(full_corrmat) <- 1
#
sims <- simulate_under_LTM(fam_vec = c("m","f"), n_fam = NULL, add_ind = TRUE,
genetic_corrmat = genetic_corrmat, full_corrmat = full_corrmat, h2 = rep(.5,3),
n_sim = 1, pop_prev = rep(.1,3))
estimate_liability(.tbl = sims$thresholds, h2 = rep(.5,3),
genetic_corrmat = genetic_corrmat, full_corrmat = full_corrmat,
pid = "indiv_ID", fam_id = "fam_ID", role = "role", out = c(1),
phen_names = paste0("phenotype", 1:3), tol = 0.01)
```

---

estimate_liability_multi

Estimating the genetic or full liability for multiple phenotypes

---

Description

estimate_liability_multi estimates the genetic component of the full liability and/or the full liability for a number of individuals based on their family history for a variable number of phenotypes.

Usage

```
estimate_liability_multi(
  .tbl = NULL,
  family_graphs = NULL,
  h2_vec,
  genetic_corrmat,
  full_corrmat,
  phen_names = NULL,
  pid = "PID",
  fam_id = "fam_ID",
  role = "role",
  family_graphs_col = "fam_graph",
  out = c(1),
  tol = 0.01
)
```

Arguments

.tbl	A matrix, list or data frame that can be converted into a tibble. Must have at least seven columns that hold the family identifier, the personal identifier, the role and the lower and upper thresholds for all phenotypes of interest. Note that the role must be one of the following abbreviations • g (Genetic component of full liability) • o (Full liability) • m (Mother) • f (Father) • c[0-9]*.[0-9]* (Children) • mgm (Maternal grandmother) • mgf (Maternal grandfather) • pgm (Paternal grandmother) • pgf (Paternal grandfather) • s[0-9]* (Full siblings) • mhs[0-9]* (Half-siblings - maternal side) • phs[0-9]* (Half-siblings - paternal side) • mau[0-9]* (Aunts/Uncles - maternal side) • pau[0-9]* (Aunts/Uncles - paternal side). Defaults to NULL.
family_graphs	A tibble with columns pid and family_graph_col. See prepare_graph for construction of the graphs. The family graphs Defaults to NULL.
h2_vec	A numeric vector representing the liability-scale heritabilities for all phenotypes. All entries in h2_vec must be non-negative and at most 1.
genetic_corrmat	A numeric matrix holding the genetic correlations between the desired phenotypes. All diagonal entries must be equal to one, while all off-diagonal entries must be between -1 and 1. In addition, the matrix must be symmetric.
full_corrmat	A numeric matrix holding the full correlations between the desired phenotypes. All diagonal entries must be equal to one, while all off-diagonal entries must be between -1 and 1. In addition, the matrix must be symmetric.
phen_names	A character vector holding the phenotype names. These names will be used to create the row and column names for the covariance matrix. If it is not specified, the names will default to phenotype1, phenotype2, etc. Defaults to NULL.
pid	A string holding the name of the column in family and threshs that hold the personal identifier(s). Defaults to "PID".
fam_id	A string holding the name of the column in family that holds the family identifier. Defaults to "fam_ID".
role	A string holding the name of the column in .tbl that holds the role.Each role must be chosen from the following list of abbreviations • g (Genetic component of full liability) • o (Full liability) • m (Mother) • f (Father)

- c[0-9]*.[0-9]* (Children)
- mgm (Maternal grandmother)
- mgf (Maternal grandfather)
- pgm (Paternal grandmother)
- pgf (Paternal grandfather)
- s[0-9]* (Full siblings)
- mhs[0-9]* (Half-siblings - maternal side)
- phs[0-9]* (Half-siblings - paternal side)
- mau[0-9]* (Aunts/Uncles - maternal side)
- pau[0-9]* (Aunts/Uncles - paternal side). Defaults to "role".

family_graphs_col
: Name of column with family graphs in family_graphs. Defaults to "fam_graph".

out
: A character or numeric vector indicating whether the genetic component of the full liability, the full liability or both should be returned. If out = c(1) or out = c("genetic"), the genetic liability is estimated and returned. If out = c(2) or out = c("full"), the full liability is estimated and returned. If out = c(1,2) or out = c("genetic", "full"), both components are estimated and returned. Defaults to c(1).

tol
: A number that is used as the convergence criterion for the Gibbs sampler. Equals the standard error of the mean. That is, a tolerance of 0.2 means that the standard error of the mean is below 0.2. Defaults to 0.01.

Details

This function can be used to estimate either the genetic component of the full liability, the full liability or both for a variable number of traits.

Value

If family and threshs are two matrices, lists or data frames that can be converted into tibbles, if family has two columns named like the strings represented in pid and fam_id, if threshs has a column named like the string given in pid as well as a column named "lower" and a column named "upper" and if the liability-scale heritabilities in h2_vec, genetic_corrmat, full_corrmat, out and tol are of the required form, then the function returns a tibble with at least six columns (depending on the length of out). The first two columns correspond to the columns fam_id and pid present in the tibble family. If out is equal to c(1) or c("genetic"), the third and fourth columns hold the estimated genetic liability as well as the corresponding standard error for the first phenotype, respectively. If out equals c(2) or c("full"), the third and fourth columns hold the estimated full liability as well as the corresponding standard error for the first phenotype, respectively. If out is equal to c(1,2) or c("genetic","full"), the third and fourth columns hold the estimated genetic liability as well as the corresponding standard error for the first phenotype, respectively, while the fifth and sixth columns hold the estimated full liability as well as the corresponding standard error for the first phenotype, respectively. The remaining columns hold the estimated genetic liabilities and/or the estimated full liabilities as well as the corresponding standard errors for the remaining phenotypes.

See Also

future_apply, estimate_liability_single, estimate_liability

Examples

```
genetic_corrmat <- matrix(0.4, 3, 3)
diag(genetic_corrmat) <- 1
full_corrmat <- matrix(0.6, 3, 3)
diag(full_corrmat) <- 1
#
sims <- simulate_under_LTM(fam_vec = c("m","f"), n_fam = NULL, add_ind = TRUE,
genetic_corrmat = genetic_corrmat, full_corrmat = full_corrmat, h2 = rep(.5,3),
n_sim = 1, pop_prev = rep(.1,3))
estimate_liability_multi(.tbl = sims$thresholds, h2_vec = rep(.5,3),
genetic_corrmat = genetic_corrmat, full_corrmat = full_corrmat,
pid = "indiv_ID", fam_id = "fam_ID", role = "role", out = c(1),
phen_names = paste0("phenotype", 1:3), tol = 0.01)
```

---

estimate_liability_single

Estimating the genetic or full liability

Description

estimate_liability_single estimates the genetic component of the full liability and/or the full liability for a number of individuals based on their family history.

Usage

```
estimate_liability_single(
  .tbl = NULL,
  family_graphs = NULL,
  h2 = 0.5,
  pid = "PID",
  fam_id = "fam_ID",
  family_graphs_col = "fam_graph",
  role = NULL,
  out = c(1),
  tol = 0.01
)
```

Arguments

.tbl	A matrix, list or data frame that can be converted into a tibble. Must have at least five columns that hold the family identifier, the personal identifier, the role and the lower and upper thresholds. Note that the role must be one of the following abbreviations

- g (Genetic component of full liability)
- o (Full liability)
- m (Mother)
- f (Father)
- c[0-9]*.[0-9]* (Children)
- mgm (Maternal grandmother)
- mgf (Maternal grandfather)
- pgm (Paternal grandmother)
- pgf (Paternal grandfather)
- s[0-9]* (Full siblings)
- mhs[0-9]* (Half-siblings - maternal side)
- phs[0-9]* (Half-siblings - paternal side)
- mau[0-9]* (Aunts/Uncles - maternal side)
- pau[0-9]* (Aunts/Uncles - paternal side). Defaults to NULL.

family_graphs A tibble with columns pid and family_graph_col. See prepare_graph for construction of the graphs. The family graphs Defaults to NULL.

h2 A number representing the heritability on liability scale for a single phenotype. Must be non-negative. Note that under the liability threshold model, the heritability must also be at most 1. Defaults to 0.5.

pid A string holding the name of the column in .tbl (or family and threshs) that hold the personal identifier(s). Defaults to "PID".

fam_id A string holding the name of the column in .tbl or family that holds the family identifier. Defaults to "fam_ID".

family_graphs_col
Name of column with family graphs in family_graphs. Defaults to "fam_graph".

role A string holding the name of the column in .tbl that holds the role. Each role must be chosen from the following list of abbreviations

- g (Genetic component of full liability)
- o (Full liability)
- m (Mother)
- f (Father)
- c[0-9]*.[0-9]* (Children)
- mgm (Maternal grandmother)
- mgf (Maternal grandfather)
- pgm (Paternal grandmother)
- pgf (Paternal grandfather)
- s[0-9]* (Full siblings)
- mhs[0-9]* (Half-siblings - maternal side)
- phs[0-9]* (Half-siblings - paternal side)
- mau[0-9]* (Aunts/Uncles - maternal side)
- pau[0-9]* (Aunts/Uncles - paternal side). Defaults to "role".

`out` A character or numeric vector indicating whether the genetic component of the full liability, the full liability or both should be returned. If `out = c(1)` or `out = c("genetic")`, the genetic liability is estimated and returned. If `out = c(2)` or `out = c("full")`, the full liability is estimated and returned. If `out = c(1,2)` or `out = c("genetic", "full")`, both components are estimated and returned. Defaults to `c(1)`.

`tol` A number that is used as the convergence criterion for the Gibbs sampler. Equals the standard error of the mean. That is, a tolerance of 0.2 means that the standard error of the mean is below 0.2. Defaults to 0.01.

Details

This function can be used to estimate either the genetic component of the full liability, the full liability or both. It is possible to input either

Value

If `family` and `threshs` are two matrices, lists or data frames that can be converted into tibbles, if `family` has two columns named like the strings represented in `pid` and `fam_id`, if `threshs` has a column named like the string given in `pid` as well as a column named "lower" and a column named "upper" and if the liability-scale heritability `h2`, `out`, `tol` and `always_add` are of the required form, then the function returns a tibble with either four or six columns (depending on the length of out). The first two columns correspond to the columns `fam_id` and `pid` ' present in `family`. If `out` is equal to `c(1)` or `c("genetic")`, the third and fourth column hold the estimated genetic liability as well as the corresponding standard error, respectively. If `out` equals `c(2)` or `c("full")`, the third and fourth column hold the estimated full liability as well as the corresponding standard error, respectively. If `out` is equal to `c(1,2)` or `c("genetic","full")`, the third and fourth column hold the estimated genetic liability as well as the corresponding standard error, respectively, while the fifth and sixth column hold the estimated full liability as well as the corresponding standard error, respectively.

See Also

`future_apply`, `estimate_liability_multi`, `estimate_liability`

Examples

```
sims <- simulate_under_LTM(fam_vec = c("m","f","s1"), n_fam = NULL,
add_ind = TRUE, h2 = 0.5, n_sim=10, pop_prev = .05)
#
estimate_liability_single(.tbl = sims$thresholds,
h2 = 0.5, pid = "indiv_ID", fam_id = "fam_ID", role = "role", out = c(1),
tol = 0.01)
#
sims <- simulate_under_LTM(fam_vec = c(), n_fam = NULL, add_ind = TRUE,
h2 = 0.5, n_sim=10, pop_prev = .05)
#
estimate_liability_single(.tbl = sims$thresholds,
h2 = 0.5, pid = "indiv_ID", fam_id = "fam_ID", role = "role",
out = c("genetic"), tol = 0.01)
```

`familywise_attach_attributes` *Wrapper to attach attributes to family graphs*

Description

This function can attach attributes to family graphs, such as lower and upper thresholds, for each family member. This allows for personalised thresholds and other per-family specific attributes. This function wraps around attach_attributes to ease the process of attaching attributes to family graphs in the standard format.

Usage

```
familywise_attach_attributes(
  family_graphs,
  fam_attr,
  fam_graph_col = "fam_graph",
  attached_fam_graph_col = "masked_fam_graph",
  fid = "fid",
  pid = "pid",
  cols_to_attach = c("lower", "upper"),
  proband_cols_to_censor = NA
)
```

Arguments

`family_graphs`	tibble with family ids and family graphs
`fam_attr`	tibble with attributes for each family member
`fam_graph_col`	column name of family graphs in family_graphs. defailts to "fam_graph"
`attached_fam_graph_col`	column name of the updated family graphs with attached attributes. defaults to "masked_fam_graph".
`fid`	column name of family id. Typically contains the name of the proband that a family graph is centred on. defaults to "fid".
`pid`	personal identifier for each individual in a family. Allows for multiple instances of the same individual across families. Defaults to "pid".
`cols_to_attach`	columns to attach to the family graphs from fam_attr, typically lower and upper thresholds. Mixture input also requires K_i and K_pop.
`proband_cols_to_censor`	Should proband's upper and lower thresholds be made uninformative? Defaults to TRUE. Used to exclude proband's information for prediction.

Value

tibble with family ids and an updated family graph with attached attributes. If lower and upper thresholds are specified, the input is ready for estimate_liability().

Examples

```
# See Vignettes.
```

`fixSexCoding` *Fixing sex coding in trio info*

Description

Internal function used to assist in fixing sex coding separately from id coding type.

Usage

```
fixSexCoding(x, sex_coding = TRUE, dadid, momid)
```

Arguments

`x`	current row to check against
`sex_coding`	logical. Is sex coded as character?
`dadid`	column name of father ids
`momid`	column name of mother ids

Value

appropriate sex coding

`get_all_combs` *construct all combinations of input vector*

Description

pastes together all combinations of input vector

Usage

```
get_all_combs(vec)
```

Arguments

`vec`	vector of strings

Value

A vector of strings is returned.

Examples

```
get_all_combs(letters[1:3])
```

`get_family_graphs` *Automatically identify family members of degree n*

Description

This function identifies individuals ndegree-steps away from the proband in the population graph.

Usage

```
get_family_graphs(
  pop_graph,
  ndegree,
  proband_vec,
  fid = "fid",
  fam_graph_col = "fam_graph",
  mindist = 0,
  mode = "all"
)
```

Arguments

`pop_graph`	Population graph from prepare_graph()
`ndegree`	Number of steps away from proband to include
`proband_vec`	Vector of proband ids to create family graphs for. Must be strings.
`fid`	Column name of proband ids in the output.
`fam_graph_col`	Column name of family graphs in the output.
`mindist`	Minimum distance from proband to exclude in the graph (experimental, untested), defaults to 0, passed directly to make_neighborhood_graph.
`mode`	Type of distance measure in the graph (experimental, untested), defaults to "all", passed directly to make_neighborhood_graph.

Value

Tibble with two columns, family ids (fid) and family graphs (fam_graph_col).

Examples

```
# See Vignettes.
```

`get_generations` *Compute Generational Distances and Kinship Coefficients from a Family Graph*

Description

Calculates generational distances and kinship coefficients between all pairs of individuals represented in a directed family graph. The function identifies shortest paths between individuals, accounts for common ancestors, and derives kinship coefficients based on the number of generations separating each pair.

Usage

```
get_generations(fam_graph)
```

Arguments

`fam_graph`	An igraph object representing the family structure, where directed edges indicate parent–child relationships (from parent to child). See get_family_graphs().

Value

A tibble with one row per unique pair of individuals and the following columns:

id1 Identifier for the first individual.

id2 Identifier for the second individual.

k Estimated kinship coefficient between `id1` and `id2`.

gen.x Mean number of generations separating `id1` from the common ancestor.

gen.y Mean number of generations separating `id2` from the common ancestor.

The family graph is centered on a proband (typically the same id as fid), all relations for the proband can be found by selecting only relations with the proband's id in the column id1.

Examples

```
# see vignette on identifying and labelling relatives
```

get_kinship *Construct kinship matrix from graph*

Description

construct the kinship matrix from a graph representation of a family, centered on an index person (proband).

Usage

```
get_kinship(fam_graph, h2, index_id = NA, add_ind = TRUE, fix_diag = TRUE)
```

Arguments

fam_graph	graph.
h2	heritability.
index_id	proband id. Only used in conjuction with add_ind = TRUE.
add_ind	add genetic liability to the kinship matrix. Defaults to true.
fix_diag	Whether to set diagonal to 1 for all entries except for the genetic liability.

Value

A kinship matrix.

Examples

```
fam <- data.frame(
i = c(1, 2, 3, 4),
f = c(3, 0, 4, 0),
m = c(2, 0, 0, 0)
)

thresholds <- data.frame(
  i = c(1, 2, 3, 4),
  lower = c(-Inf, -Inf, 0.8, 0.7),
  upper = c(0.8, 0.8, 0.8, 0.7)
)

graph <- prepare_graph(fam, icol = "i", fcol = "f", mcol = "m", node_attributes = thresholds)

get_kinship(graph, h2 = 0.5, index_id = "1")
get_kinship(graph, h2 = 1, add_ind = FALSE)
```

`get_relatedness` *Relatedness between a pair of family members*

Description

`get_relatedness` returns the relatedness times the liability-scale heritability for a pair of family members

Usage

```
get_relatedness(s1, s2, h2 = 0.5, from_covmat = FALSE)
```

Arguments

`s1`, `s2`	Strings representing the two family members. The strings must be chosen from the following list of strings: • `g` (Genetic component of full liability) • `o` (Full liability) • `m` (Mother) • `f` (Father) • `c[0-9]*.[0-9]*` (Children) • `mgm` (Maternal grandmother) • `mgf` (Maternal grandfather) • `pgm` (Paternal grandmother) • `pgf` (Paternal grandfather) • `s[0-9]*` (Full siblings) • `mhs[0-9]*` (Half-siblings - maternal side) • `phs[0-9]*` (Half-siblings - paternal side) • `mau[0-9]*` (Aunts/Uncles - maternal side) • `pau[0-9]*` (Aunts/Uncles - paternal side).
`h2`	A number representing the squared heritability on liability scale. Must be non-negative and at most 1. Defaults to 0.5
`from_covmat`	logical variable. Only used internally. allows for skip of negative check.

Details

This function can be used to get the percentage of shared DNA times the liability-scale heritability h^2 for two family members.

Value

If both `s1` and `s2` are strings chosen from the mentioned list of strings and `h2` is a number satisfying $0 \leq h2 \leq 1$, then the output will be a number that equals the percentage of shared DNA between `s1` and `s2` times the squared heritability `h2`.

Note

If you are only interested in the percentage of shared DNA, set `h2 = 1`.

Examples

```
get_relatedness("g","o")
get_relatedness("g","f", h2 = 1)
get_relatedness("o","s", h2 = 0.3)


# This will result in errors:
try(get_relatedness("a","b"))
try(get_relatedness(m, mhs))
```

---

get_relations *Compute and Label Pairwise Relationships Across Multiple Family Graphs*

Description

Applies `get_generations()` and `label_relatives()` to a tibble of family graph objects from `get_family_graphs()`, returning a unified table of labelled pairwise relationships for all individuals across the specified families.

Usage

```
get_relations(
  family_graphs,
  fid = "fid",
  family_id_vec = NULL,
  fam_graph_col = "fam_graph"
)
```

Arguments

`family_graphs`	A tibble containing family-specific graph objects from `get_family_graphs()` (typically of class `igraph`). Each row should correspond to a distinct family, with one column containing the graph object and another containing the family identifier (typically the proband's id).
`fid`	A character string specifying the name of the column in `family_graphs` that holds the family identifiers. Defaults to `"fid"`.
`family_id_vec`	An optional character or numeric vector specifying which families to process. If `NULL` (default), the function will process all families in `family_graphs`.
`fam_graph_col`	A character string specifying the name of the column in `family_graphs` that contains the family graph objects. Defaults to `"fam_graph"`.

Value

A tibble containing all labelled pairwise relationships across the specified families, with columns:

fid Family identifier (typically the proband's id).

id1, id2 Identifiers for the two individuals being compared.

gen.x, gen.y Number of generations separating each individual from their most recent common ancestor.

k Kinship coefficient between the pair.

lab Relationship label (e.g., `"S"`, `"P"`, `"GP"`, `"1C"`, `"HNib"`).

See Also

`get_generations()` for computing generational distances and kinship coefficients, `label_relatives()` for labelling relationships based on generational patterns.

Examples

```
# See vignette
```

---

graph_based_covariance_construction

Constructing covariance matrix from local family graph

Description

Function that constructs the genetic covariance matrix given a graph around a proband and extracts the threshold information from the graph.

Usage

```
graph_based_covariance_construction(
  pid,
  cur_proband_id,
  cur_family_graph,
  h2,
  add_ind = TRUE
)
```

Arguments

`pid`	Name of column of personal ID
`cur_proband_id`	id of proband
`cur_family_graph`	local graph of current proband
`h2`	liability scale heritability
`add_ind`	whether to add genetic liability of the proband or not. Defaults to true.

Value

list with two elements. The first element is temp_tbl, which contains the id of the current proband, the family ID and the lower and upper thresholds. The second element, cov, is the covariance matrix of the local graph centered on the current proband.

Examples

```
fam <- data.frame(
  id = c("pid", "mom", "dad", "pgf"),
  dadcol = c("dad", 0, "pgf", 0),
  momcol = c("mom", 0, 0, 0))

thresholds <- data.frame(
  id = c("pid", "mom", "dad", "pgf"),
  lower = c(-Inf, -Inf, 0.8, 0.7),
  upper = c(0.8, 0.8, 0.8, 0.7))

graph <- prepare_graph(fam, icol = "id", fcol = "dadcol", mcol = "momcol",
 node_attributes = thresholds)

graph_based_covariance_construction(pid = "id",
                                    cur_proband_id = "pid",
                                    cur_family_graph = graph,
                                    h2 = 0.5)
```

---

graph_based_covariance_construction_multi

Constructing covariance matrix from local family graph for multi trait analysis

---

Description

Function that constructs the genetic covariance matrix given a graph around a proband and extracts the threshold information from the graph.

Usage

```
graph_based_covariance_construction_multi(
  fam_id,
  pid,
  cur_proband_id,
  cur_family_graph,
  h2_vec,
  genetic_corrmat,
  phen_names,
  add_ind = TRUE
)
```

Arguments

fam_id	Name of column with the family ID
pid	Name of column of personal ID
cur_proband_id	id of proband
cur_family_graph	local graph of current proband
h2_vec	vector of liability scale heritabilities
genetic_corrmat	matrix with genetic correlations between considered phenotypes. Must have same order as h2_vec.
phen_names	Names of the phenotypes, as given in cur_family_graph.
add_ind	whether to add genetic liability of the proband or not. Defaults to true.

Value

list with three elements. The first element is temp_tbl, which contains the id of the current proband, the family ID and the lower and upper thresholds for all phenotypes. The second element, cov, is the covariance matrix of the local graph centred on the current proband. The third element is newOrder, which is the order of ids from pid and phen_names pasted together, such that order can be enforced elsewhere too.

Examples

```
fam <- data.frame(
fam = c(1, 1, 1,1),
id = c("pid", "mom", "dad", "pgf"),
dadcol = c("dad", 0, "pgf", 0),
momcol = c("mom", 0, 0, 0))

thresholds <- data.frame(
  id = c("pid", "mom", "dad", "pgf"),
  lower_1 = c(-Inf, -Inf, 0.8, 0.7),
  upper_1 = c(0.8, 0.8, 0.8, 0.7),
  lower_2 = c(-Inf, 0.3, -Inf, 0.2),
  upper_2 = c(0.3, 0.3, 0.3, 0.2))

graph <- prepare_graph(fam, icol = "id", fcol = "dadcol", mcol = "momcol",
 node_attributes = thresholds)

ntrait <- 2
genetic_corrmat <- matrix(0.2, ncol = ntrait, nrow = ntrait)
diag(genetic_corrmat) <- 1
full_corrmat <- matrix(0.3, ncol = ntrait, nrow = ntrait)
diag(full_corrmat) <- 1
h2_vec <- rep(0.6, ntrait)

graph_based_covariance_construction_multi(fam_id = "fam",
                                          pid = "id",
                                          cur_proband_id = "pid",
```

```
cur_family_graph = graph,
h2_vec = h2_vec,
genetic_corrmat = genetic_corrmat,
phen_names = c("1", "2"))
```

`graph_to_trio` *Convert from igraph to trio information*

Description

This function converts an igraph object to a trio information format.

Usage

```
graph_to_trio(
  graph,
  id = "id",
  dadid = "dadid",
  momid = "momid",
  sex = "sex",
  fixParents = TRUE
)
```

Arguments

`graph`	An igraph graph object.
`id`	Column of proband id. Defaults to id.
`dadid`	Column of father id. Defaults to dadid.
`momid`	Column of mother id. Defaults to momid.
`sex`	Column of sex in igraph attributes. Defaults to sex.
`fixParents`	Logical. If TRUE, the kinship2's fixParents will be run on the trio information before returning. Defaults to TRUE.

Details

The sex column is required in the igraph attributes. The sex information is used to determine who is the mother and father in the trio.

Value

A tibble with trio information.

Examples

```
if (FALSE) {

family = tribble(
~id, ~momcol, ~dadcol,
"pid", "mom", "dad",
"sib", "mom", "dad",
"mhs", "mom", "dad2",
"phs", "mom2", "dad",
"mom", "mgm", "mgf",
"dad", "pgm", "pgf",
"dad2", "pgm2", "pgf2",
"paunt", "pgm", "pgf",
"pacousin", "paunt", "pauntH",
"hspaunt", "pgm", "newpgf",
"hspacousin", "hspaunt", "hspauntH",
"puncle", "pgm", "pgf",
"pucousin", "puncleW", "puncle",
"maunt", "mgm", "mgf",
"macousin", "maunt", "mauntH",
"hsmuncle", "newmgm", "mgf",
"hsmucousin", "hsmuncleW", "hsmuncle"
)


thrs =  tibble(
  id = family %>% select(1:3) %>% unlist() %>% unique(),
 sex = case_when(
   id %in% family$momcol ~ "F",
   id %in% family$dadcol ~ "M",
    TRUE ~ NA)) %>%
  mutate(sex = sapply(sex, function(x) ifelse(is.na(x),
  sample(c("M", "F"), 1), x)))
graph = prepare_graph(.tbl = family,
icol = "id", fcol = "dadcol", mcol = "momcol", node_attributes = thrs)
graph_to_trio(graph)
}
```

`label_relatives` *Label Pairwise Relationships Based on Generational Distance and Kinship Coefficient*

Description

Assigns standard pedigree relationship labels (e.g., *Parent*, *Child*, *Sibling*, *Grandparent*, *Cousin*) to all pairs of individuals based on their generational distances (`gen.x`, `gen.y`) and kinship coefficients (`k`), typically produced by `get_generations()`.

Usage

```
label_relatives(tbl)
```

Arguments

`tbl`	A tibble or data frame containing at least the following columns: **fid** Column with family identifier (typically the proband's id). **id1** Identifier for the first individual. **id2** Identifier for the second individual. **gen.x** Number of generations between `id1` and their most recent common ancestor with `id2`. **gen.y** Number of generations between `id2` and their most recent common ancestor with `id1`. **k** Estimated kinship coefficient between the two individuals.

Details

This function derives descriptive relationship labels using generational differences and kinship patterns. The labels are written in a short-hand notation, an explaination of a subset is given below:

- *P* - Parent
- *Ch* - Child
- *S* - Sibling
- *GP* - Grandparent
- *Pib* - "Pibling" (parental sibling; aunt/uncle)
- *Nib* - "Nibling" (sibling's child; niece/nephew)
- *GCh* - Grandchild
- *GPib* - Grandpibling (grandparent's sibling)
- *GNib* - Grandnibling (sibling's grandchild)
- *C* - Cousin
- *1C1R* - First Cousin Once Removed
- *2C2R* - Second Cousin Twice Removed
- *H* prefix - Half relationships (e.g., *HS* for Half-Sibling)

Value

A tibble with the following columns:

fid Column with family identifier (typically the proband's id).

id1 Identifier for the first individual.

id2 Identifier for the second individual.

gen.x Generational distance for `id1`.

gen.y Generational distance for `id2`.

k Kinship coefficient between the two individuals.

lab Assigned relationship label (e.g., `"S"`, `"P"`, `"1C"`, `"H1C"`, `"2GP"`, etc.).

See Also

`get_generations()` for computing the generational and kinship inputs used by this function.

Examples

```
# see vignette on identifying and labelling relatives
```

`prepare_graph` *Construct graph from register information*

Description

`prepare_graph` constructs a graph based on mother, father, and offspring links.

Usage

```
prepare_graph(
  .tbl,
  icol,
  fcol,
  mcol,
  node_attributes = NA,
  missingID_patterns = "^0$"
)
```

Arguments

`.tbl`	tibble with columns icol, fcol, mcol. Additional columns will be attributes in the constructed graph.
`icol`	column name of column with proband ids.
`fcol`	column name of column with father ids.
`mcol`	column name of column with mother ids.
`node_attributes`	tibble with icol and any additional information, such as sex, lower threshold, and upper threshold. Used to assign attributes to each node in the graph, e.g. lower and upper thresholds to individuals in the graph.
`missingID_patterns`	string of missing values in the ID columns. Multiple values can be used, but must be separated by "|". Defaults to "^0$". OBS: "0" is NOT enough, since it relies on regex.

Value

An igraph object. A (directed) graph object based on the links provided in .tbl, potentially with provided attributes stored for each node.

Examples

```
fam <- data.frame(
  id = c("pid", "mom", "dad", "pgf"),
  dadcol = c("dad", 0, "pgf", 0),
  momcol = c("mom", 0, 0, 0))

thresholds <- data.frame(
  id = c("pid", "mom", "dad", "pgf"),
  lower = c(-Inf, -Inf, 0.8, 0.7),
  upper = c(0.8, 0.8, 0.8, 0.7))

prepare_graph(fam, icol = "id", fcol = "dadcol", mcol = "momcol", node_attributes = thresholds)
```

---

prepare_LTFHPlus_input

Prepares input for `estimate_liability`

Description

Prepares input for `estimate_liability`

Usage

```
prepare_LTFHPlus_input(
  .tbl,
  CIP,
  age_col,
  aoo_col,
  CIP_merge_columns = c("sex", "birth_year", "age"),
  CIP_cip_col = "cip",
  status_col = "status",
  use_fixed_case_thr = FALSE,
  fam_id_col = "fam_id",
  personal_id_col = "pid",
  interpolation = NULL,
 bst.params = list(max_depth = 10, base_score = 0, nthread = 4, min_child_weight = 10),
  min_CIP_value = 1e-05,
  xgboost_itr = 50
)
```

Arguments

`.tbl`	contains family and personal ids and role with a family.
`CIP`	tibble with population representative cumulative incidence proportions. CIP values should be merged by `CIP_columns`.
`age_col`	name of column with age

aoo_col	name of column with age of onset
CIP_merge_columns	The columns the CIPs are subset by, e.g. CIPs by birth_year, sex.
CIP_cip_col	name of column with CIP values
status_col	Column that contains the status of each family member
use_fixed_case_thr	Should the threshold be fixed for cases? Can be used if CIPs are detailed, e.g. stratified by birth_year and sex.
fam_id_col	Column that contains the family ID
personal_id_col	Column that contains the personal ID
interpolation	type of interpolation, defaults to NULL.
bst.params	list of parameters to pass on to xgboost
min_CIP_value	minimum cip value to allow, too low values may lead to numerical instabilities.
xgboost_itr	Number of iterations to run xgboost for.

Value

tibble formatted for `estimate_liability`

Examples

```
tbl = data.frame(
  fam_id = c(1, 1, 1, 1),
  pid = c(1, 2, 3, 4),
  role = c("o", "m", "f", "pgf"),
  sex = c(1, 0, 1, 1),
  status = c(0, 0, 1, 1),
  age = c(22, 42, 48, 78),
  birth_year = 2023 - c(22, 42, 48, 78),
  aoo = c(NA, NA, 43, 45))

cip = data.frame(
  age = c(22, 42, 43, 45, 48, 78),
  birth_year = c(2001, 1981, 1975, 1945, 1975, 1945),
  sex = c(1, 0, 1, 1, 1, 1),
  cip = c(0.1, 0.2, 0.3, 0.3, 0.3, 0.4))

prepare_LTFHPlus_input(.tbl = tbl,
                       CIP = cip,
                       age_col = "age",
                       aoo_col = "aoo",
                       interpolation = NA)
```

`Relation_per_proband_plot` *Plot the (Average) Number of Relatives per Proband by Relationship Type*

Description

Produces a structured visualisation of the (average) number of each relationship type per proband, based on the labelled pairwise relationship data from `label_relatives()`. The plot arranges relationship types according to generational distance and degree of relatedness, providing an intuitive overview of kinship structure within the study sample.

Usage

```
Relation_per_proband_plot(
  labelled_relations,
  proband_vec,
  reported_info = "both"
)
```

Arguments

`labelled_relations`
: A tibble or data frame containing pairwise relationship labels and their associated metadata. Must include the following columns:

  id1 Identifier for the first individual (typically the proband).

  id2 Identifier for the second individual (the relative).

  gen.x, gen.y Number of generations separating each individual from their most recent common ancestor.

  k Kinship coefficient between the two individuals.

  lab Relationship label assigned by `label_relatives()`.

`proband_vec`
: A vector of identifiers for probands. The function restricts the analysis to pairs where `id1` is included in this vector.

`reported_info`
: Chose which information is reported on the figure.

  total shows the total number of relatives of each type across all probands.

  average shows the average number of relatives of each type per proband.

  both (default) shows both total and average numbers on the plot.

Details

If any relationship types in the input are not recognised in the predefined mapping (e.g., rare or complex kinships), these are aggregated and shown as `"Other"`.

Value

A **ggplot2** object showing the (mean) number of relatives per proband for each relationship type. The plot can be further modified using standard **ggplot2** functions (e.g., `+ theme()` or `+ labs()`).

See Also

`label_relatives()` for generating the relationship labels used as input.

Examples

```
# see vignette on identifying and labelling relatives
```

---

rtmvnorm.gibbs *Gibbs Sampler for the truncated multivariate normal distribution*

Description

`rtmvnorm.gibbs` implements Gibbs sampler for the truncated multivariate normal distribution with covariance matrix `covmat`.

Usage

```
rtmvnorm.gibbs(
  n_sim = 1e+05,
  covmat,
  lower = -Inf,
  upper,
  fixed = (lower == upper),
  out = c(1),
  burn_in = 1000
)
```

Arguments

`n_sim`	A positive number representing the number of draws from the Gibbs sampler after burn-in.. Defaults to `1e+05`.
`covmat`	A symmetric and numeric matrix representing the covariance matrix for the multivariate normal distribution.
`lower`	A number or numeric vector representing the lower cutoff point(s) for the truncated normal distribution. The length of lower must be 1 or equal to the dimension of the multivariable normal distribution. Defaults to `-Inf`.
`upper`	A number or numeric vector representing the upper cutoff point(s) for the truncated normal distribution. Must be greater or equal to lower. In addition the length of upper must be 1 or equal to the dimension of the multivariable normal distribution. Defaults to `Inf`.

fixed	A logical scalar or a logical vector indicating which variables to fix. If fixed is a vector, it must have the same length as lower and upper. Defaults to TRUE when lower is equal to upper and FALSE otherwise.
out	An integer or numeric vector indicating which variables should be returned from the Gibbs sampler. If out = c(1), the first variable (usually the genetic component of the full liability of the first phenotype) is estimated and returned. If out = c(2), the second variable (usually full liability) is estimated and returned. If out = c(1,2), both the first and the second variable are estimated and returned. Defaults to c(1).
burn_in	A number of iterations that count as burn in for the Gibbs sampler. Must be non-negative. Defaults to 1000.

Details

Given a covariance matrix covmat and lower and upper cutoff points, the function rtmvnorm.gibbs() can be used to perform Gibbs sampler on a truncated multivariable normal distribution. It is possible to specify which variables to return from the Gibbs sampler, making it convenient to use when estimating only the full liability or the genetic component of the full liability.

Value

If covmat is a symmetric and numeric matrix, if n_sim and burn_in are positive/non-negative numbers, if out is a numeric vector and lower, upper and fixed are numbers or vectors of the same length and the required format, rtmvnorm.gibbs returns the sampling values from the Gibbs sampler for all variables specified in out.

References

Kotecha, J. H., & Djuric, P. M. (1999, March). Gibbs sampling approach for generation of truncated multivariate gaussian random variables. In 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258) (Vol. 3, pp. 1757-1760). IEEE. doi:10.1109/ICASSP.1999.756335

Wilhelm, S., & Manjunath, B. G. (2010). tmvtnorm: A package for the truncated multivariate normal distribution. The R Journal. doi:10.32614/RJ2010005

Examples

```
samp <- rtmvnorm.gibbs(10e3, covmat = matrix(c(1, 0.2, 0.2, 0.5), 2),
                       lower = c(-Inf, 0), upper = c(0, Inf), out = 1:2)
```

---

simulate_under_LTM *Simulate under the liability threshold model.*

Description

simulate_under_LTM simulates families and thresholds under the liability threshold model for a given family structure and a variable number of phenotypes.Please note that it is not possible to simulate different family structures.

Usage

```
simulate_under_LTM(
  fam_vec = c("m", "f", "s1", "mgm", "mgf", "pgm", "pgf"),
  n_fam = NULL,
  add_ind = TRUE,
  h2 = 0.5,
  genetic_corrmat = NULL,
  full_corrmat = NULL,
  phen_names = NULL,
  n_sim = 1000,
  pop_prev = 0.1
)
```

Arguments

fam_vec	A vector of strings holding the different family members. All family members must be represented by strings from the following list: • m (Mother) • f (Father) • c[0-9]*.[0-9]* (Children) • mgm (Maternal grandmother) • mgf (Maternal grandfather) • pgm (Paternal grandmother) • pgf (Paternal grandfather) • s[0-9]* (Full siblings) • mhs[0-9]* (Half-siblings - maternal side) • phs[0-9]* (Half-siblings - paternal side) • mau[0-9]* (Aunts/Uncles - maternal side) • pau[0-9]* (Aunts/Uncles - paternal side). Defaults to c("m","f","s1","mgm","mgf","pgm","pgf
n_fam	A named vector holding the desired number of family members. See setNames. All names must be picked from the list mentioned above. Defaults to NULL.
add_ind	A logical scalar indicating whether the genetic component of the full liability as well as the full liability for the underlying target individual should be included in the covariance matrix. Defaults to TRUE.
h2	Either a number or a numeric vector holding the liability-scale heritability(ies) for one or more phenotypes. All entries in h2 must be non-negative. Note that under the liability threshold model, the heritabilities must also be at most 1. Defaults to 0.5.
genetic_corrmat	Either NULL or a numeric matrix holding the genetic correlations between the desired phenotypes. Must be specified, if length(h2)>0, and will be ignored if h2 is a number. All diagonal entries in genetic_corrmat must be equal to one, while all off-diagonal entries must be between -1 and 1. In addition, the matrix must be symmetric. Defaults to NULL.

full_corrmat — Either NULL or a numeric matrix holding the full correlations between the desired phenotypes. Must be specified, if length(h2)> 0, and will be ignored if h2 is a number. All diagonal entries in full_corrmat must be equal to one, while all off-diagonal entries must be between -1 and 1. In addition, the matrix must be symmetric. Defaults to NULL.

phen_names — Either NULL or character vector holding the phenotype names. These names will be used to create the row and column names for the covariance matrix. Must be specified, if length(h2) > 0, and will be ignored if h2 is a number. If it is not specified, the names will default to phenotype1, phenotype2, etc. Defaults to NULL.

n_sim — A positive number representing the number of simulations. Defaults to 1000.

pop_prev — Either a number or a numeric vector holding the population prevalence(s), i.e. the overall prevalence(s) in the population. All entries in pop_prev must be positive and smaller than 1. Defaults to 0.1.

Details

This function can be used to simulate the case-control status, the current age and age-of-onset as well as the lower and upper thresholds for a variable number of phenotypes for all family members in each of the n_sim families. If h2 is a number, simulate_under_LTM simulates the case- control status, the current age and age-of-onset as well as thresholds for a single phenotype. However, if h2 is a numeric vector, if genetic_corrmat and full_corrmat are two symmetric correlation matrices, and if phen_names and pop_prev are to numeric vectors holding the phenotype names and the population prevalences, respectively, then simulate_under_LTM simulates the case-control status, the current age and age-of-onset as well as thresholds for two or more (correlated) phenotypes. The family members can be specified using one of two possible formats.

Value

If either fam_vec or n_fam is used as the argument, if it is of the required format, if the liability-scale heritability h2 is a number satisfying $0 \leq h^2$, n_sim is a strictly positive number, and pop_prev is a positive number that is at most one, then the output will be a list containing two tibbles. The first tibble, sim_obs, holds the simulated liabilities, the disease status and the current age/age-of-onset for all family members in each of the n_sim families. The second tibble, thresholds, holds the family identifier, the personal identifier, the role (specified in fam_vec or n_fam) as well as the lower and upper thresholds for all individuals in all families. Note that this tibble has the format required in estimate_liability. If either fam_vec or n_fam is used as the argument and if it is of the required format, if genetic_corrmat and full_corrmat are two numeric and symmetric matrices satisfying that all diagonal entries are one and that all off-diagonal entries are between -1 and 1, if the liability-scale heritabilities in h2_vec are numbers satisfying $0 \leq h_i^2$ for all $i \in \{1, ..., n_pheno\}$, n_sim is a strictly positive number, and pop_prev is a positive numeric vector such that all entries are at most one, then the output will be a list containing the following lists. The first outer list, which is named after the first phenotype in phen_names, holds the tibble sim_obs, which holds the simulated liabilities, the disease status and the current age/age-of-onset for all family members in each of the n_sim families for the first phenotype. As the first outer list, the second outer list, which is named after the second phenotype in phen_names, holds the tibble sim_obs, which holds the simulated liabilities, the disease status and the current age/age-of-onset for all family members in each of the n_sim families for the second phenotype. There is a list containing sim_obs for each

phenotype in phen_names. The last list entry, thresholds, holds the family identifier, the personal identifier, the role (specified in fam_vec or n_fam) as well as the lower and upper thresholds for all individuals in all families and all phenotypes. Note that this tibble has the format required in estimate_liability. Finally, note that if neither fam_vec nor n_fam are specified, the function returns the disease status, the current age/age-of-onset, the lower and upper thresholds, as well as the personal identifier for a single individual, namely the individual under consideration (called o). If both fam_vec and n_fam are defined, the user is asked to ' decide on which of the two vectors to use.

See Also

construct_covmat simulate_under_LTM_single simulate_under_LTM_multi

Examples

```
simulate_under_LTM()

genetic_corrmat <- matrix(0.4, 3, 3)
diag(genetic_corrmat) <- 1
full_corrmat <- matrix(0.6, 3, 3)
diag(full_corrmat) <- 1

simulate_under_LTM(fam_vec = NULL, n_fam = stats::setNames(c(1,1,1,2,2),
c("m","mgm","mgf","s","mhs")))

simulate_under_LTM(fam_vec = c("m","f","s1"), n_fam = NULL, add_ind = FALSE,
genetic_corrmat = genetic_corrmat, full_corrmat = full_corrmat, n_sim = 200)

simulate_under_LTM(fam_vec = c(), n_fam = NULL, add_ind = TRUE, h2 = 0.5,
n_sim = 200, pop_prev = 0.05)
```

---

simulate_under_LTM_multi

Simulate under the liability threshold model (multiple phenotypes).

Description

simulate_under_LTM_multi simulates families and thresholds under the liability threshold model for a given family structure and multiple phenotypes. Please note that it is not possible to simulate different family structures.

Usage

```
simulate_under_LTM_multi(
  fam_vec = c("m", "f", "s1", "mgm", "mgf", "pgm", "pgf"),
  n_fam = NULL,
  add_ind = TRUE,
  genetic_corrmat = diag(3),
```

```
  full_corrmat = diag(3),
  h2_vec = rep(0.5, 3),
  phen_names = NULL,
  n_sim = 1000,
  pop_prev = rep(0.1, 3)
)
```

Arguments

fam_vec	A vector of strings holding the different family members. All family members must be represented by strings from the following list: • m (Mother) • f (Father) • c[0-9]*.[0-9]* (Children) • mgm (Maternal grandmother) • mgf (Maternal grandfather) • pgm (Paternal grandmother) • pgf (Paternal grandfather) • s[0-9]* (Full siblings) • mhs[0-9]* (Half-siblings - maternal side) • phs[0-9]* (Half-siblings - paternal side) • mau[0-9]* (Aunts/Uncles - maternal side) • pau[0-9]* (Aunts/Uncles - paternal side). Defaults to c("m","f","s1","mgm","mgf","pgm","pgf
n_fam	A named vector holding the desired number of family members. See setNames. All names must be picked from the list mentioned above. Defaults to NULL.
add_ind	A logical scalar indicating whether the genetic component of the full liability as well as the full liability for the underlying target individual should be included in the covariance matrix. Defaults to TRUE.
genetic_corrmat	A numeric matrix holding the genetic correlations between the desired phenotypes. All diagonal entries must be equal to one, while all off-diagonal entries must be between -1 and 1. In addition, the matrix must be symmetric. Defaults to diag(3).
full_corrmat	A numeric matrix holding the full correlations between the desired phenotypes. All diagonal entries must be equal to one, while all off-diagonal entries must be between -1 and 1. In addition, the matrix must be symmetric. Defaults to diag(3).
h2_vec	A numeric vector holding the liability-scale heritabilities for a number of phenotype. All entries must be non-negative. Note that under the liability threshold model, the heritabilities must also be at most 1. Defaults to rep(0.5,3).
phen_names	A character vector holding the phenotype names. These names will be used to create the row and column names for the covariance matrix. If it is not specified, the names will default to phenotype1, phenotype2, etc. Defaults to NULL.
n_sim	A positive number representing the number of simulations. Defaults to 1000.

pop_prev A numeric vector holding the population prevalences, i.e. the overall prevalences in the population. All entries in pop_prev must be positive and smaller than 1. Defaults to rep(.1,3).

Value

If either fam_vec or n_fam is used as the argument and if it is of the required format, if genetic_corrmat and full_corrmat are two numeric and symmetric matrices satisfying that all diagonal entries are one and that all off-diagonal entries are between -1 and 1, if the liability-scale heritabilities in h2_vec are numbers satisfying $0 \leq h_i^2$ for all $i \in \{1, ..., n_pheno\}$, n_sim is a strictly positive number, and pop_prev is a positive numeric vector such that all entries are at most one, then the output will be a list containing lists for each phenotype. The first outer list, which is named after the first phenotype in phen_names, holds the tibble sim_obs, which holds the simulated liabilities, the disease status and the current age/age-of-onset for all family members in each of the n_sim families for the first phenotype. As the first outer list, the second outer list, which is named after the second phenotype in phen_names, holds the tibble sim_obs, which holds the simulated liabilities, the disease status and the current age/age-of-onset for all family members in each of the n_sim families for the second phenotype. There is a list containing sim_obs for each phenotype in phen_names. The last list entry, thresholds, holds the family identifier, the personal identifier, the role (specified in fam_vec or n_fam) as well as the lower and upper thresholds for all individuals in all families and all phenotypes. Note that this tibble has the format required in estimate_liability. Finally, note that if neither fam_vec nor n_fam are specified, the function returns the disease status, the current age/age-of-onset, the lower and upper thresholds, as well as the personal identifier for a single individual, namely the individual under consideration (called o). If both fam_vec and n_fam are defined, the user is asked to ' decide on which of the two vectors to use.

See Also

construct_covmat

Examples

```
simulate_under_LTM_multi()

genetic_corrmat <- matrix(0.4, 3, 3)
diag(genetic_corrmat) <- 1
full_corrmat <- matrix(0.6, 3, 3)
diag(full_corrmat) <- 1

simulate_under_LTM_multi(fam_vec = NULL, n_fam = stats::setNames(c(1,1,1,2,2),
c("m","mgm","mgf","s","mhs")))

simulate_under_LTM_multi(fam_vec = c("m","f","s1"), add_ind = FALSE,
genetic_corrmat = genetic_corrmat, full_corrmat = full_corrmat, n_sim = 100)

simulate_under_LTM_multi(fam_vec = c(), n_fam = NULL, add_ind = TRUE, n_sim = 150)
```

simulate_under_LTM_single *Simulate under the liability threshold model (single phenotype).*

Description

simulate_under_LTM_single simulates families and thresholds under the liability threshold model for a given family structure and a single phenotype. Please note that it is not possible to simulate different family structures.

Usage

```
simulate_under_LTM_single(
  fam_vec = c("m", "f", "s1", "mgm", "mgf", "pgm", "pgf"),
  n_fam = NULL,
  add_ind = TRUE,
  h2 = 0.5,
  n_sim = 1000,
  pop_prev = 0.1
)
```

Arguments

fam_vec	A vector of strings holding the different family members. All family members must be represented by strings from the following list: • m (Mother) • f (Father) • c[0-9]*.[0-9]* (Children) • mgm (Maternal grandmother) • mgf (Maternal grandfather) • pgm (Paternal grandmother) • pgf (Paternal grandfather) • s[0-9]* (Full siblings) • mhs[0-9]* (Half-siblings - maternal side) • phs[0-9]* (Half-siblings - paternal side) • mau[0-9]* (Aunts/Uncles - maternal side) • pau[0-9]* (Aunts/Uncles - paternal side). Defaults to c("m","f","s1","mgm","mgf","pgm","pgf
n_fam	A named vector holding the desired number of family members. See setNames. All names must be picked from the list mentioned above. Defaults to NULL.
add_ind	A logical scalar indicating whether the genetic component of the full liability as well as the full liability for the underlying target individual should be included in the covariance matrix. Defaults to TRUE.
h2	A number representing the liability-scale heritability for a single phenotype. Must be non-negative. Note that under the liability threshold model, the heritability must also be at most 1. Defaults to 0.5.

n_sim	A positive number representing the number of simulations. Defaults to 1000.
pop_prev	A positive number representing the population prevalence, i.e. the overall prevalence in the population. Must be smaller than 1. Defaults to 0.1.

Value

If either fam_vec or n_fam is used as the argument, if it is of the required format, if the liability-scale heritability h2 is a number satisfying $0 \leq h^2$, n_sim is a strictly positive number, and pop_prev is a positive number that is at most one, then the output will be a list holding two tibbles. The first tibble, sim_obs, holds the simulated liabilities, the disease status and the current age/age-of-onset for all family members in each of the n_sim families. The second tibble, thresholds, holds the family identifier, the personal identifier, the role (specified in fam_vec or n_fam) as well as the lower and upper thresholds for all individuals in all families. Note that this tibble has the format required in estimate_liability. In addition, note that if neither fam_vec nor n_fam are specified, the function returns the disease status, the current age/age-of-onset, the lower and upper thresholds, as well as the personal identifier for a single individual, namely the individual under consideration (called o). If both fam_vec and n_fam are defined, the user is asked to ' decide on which of the two vectors to use.

See Also

construct_covmat, simulate_under_LTM_multi, simulate_under_LTM

Examples

```
simulate_under_LTM_single()
simulate_under_LTM_single(fam_vec = NULL, n_fam = stats::setNames(c(1,1,1,2),
c("m","mgm","mgf","mhs")))
simulate_under_LTM_single(fam_vec = c("m","f","s1"), n_fam = NULL, add_ind = FALSE,
h2 = 0.5, n_sim = 500, pop_prev = .05)
simulate_under_LTM_single(fam_vec = c(), n_fam = NULL, add_ind = TRUE, h2 = 0.5,
n_sim = 200, pop_prev = 0.05)
```

---

truncated_normal_cdf *CDF for truncated normal distribution.*

Description

truncated_normal_cdf computes the cumulative density function for a truncated normal distribution.

Usage

```
truncated_normal_cdf(
  liability,
  lower = stats::qnorm(0.05, lower.tail = FALSE),
  upper = Inf
)
```

Arguments

liability	A number representing the individual's true underlying liability.
lower	A number representing the lower cutoff point for the truncated normal distribution. Defaults to 1.645 (stats::qnorm(0.05, lower.tail = FALSE)).
upper	A number representing the upper cutoff point of the truncated normal distribution. Must be greater or equal to lower. Defaults to Inf.

Details

This function can be used to compute the value of the cumulative density function for a truncated normal distribution given an individual's true underlying liability.

Value

If liability is a number and the lower and upper cutoff points are numbers satisfying lower <= upper, then `truncated_normal_cdf` returns the probability that the liability will take on a value less than or equal to `liability`.

Examples

```
curve(sapply(liability, truncated_normal_cdf), from = qnorm(0.05, lower.tail = FALSE), to = 3.5,
 xname = "liability")
```

Index