

# Package ‘LTFGRS’

August 29, 2025

**Type** Package

**Title** Implementation of Several Phenotype-Based Family Genetic Risk Scores

**Version** 1.0.1

**Description** Implementation of several phenotype-based family genetic risk scores with unified input data and data preparation functions to help facilitate the required data preparation and management. The implemented family genetic risk scores are the extended liability threshold model conditional on family history from Pedersen (2022) <[doi:10.1016/j.ajhg.2022.01.009](https://doi.org/10.1016/j.ajhg.2022.01.009)> and Pedersen (2023) <<https://www.nature.com/articles/s41467-023-41210-z>>, Pearson-Aitken Family Genetic Risk Scores from Krebs (2024) <[doi:10.1016/j.ajhg.2024.09.009](https://doi.org/10.1016/j.ajhg.2024.09.009)>, and family genetic risk score from Kendler (2021) <[doi:10.1001/jamapsychiatry.2021.0336](https://doi.org/10.1001/jamapsychiatry.2021.0336)>.

**Imports** batchmeans, dplyr, future.apply, future, lubridate, purrr, Rcpp, rlang, stats, stringr, tibble, tmvtnorm, tidyselect, igraph, xgboost, tidy

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0), MASS

**LinkingTo** Rcpp

**Config/testthat/edition** 3

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**Language** en-GB

**License** GPL (>= 3)

**Encoding** UTF-8

**URL** <https://emilmip.github.io/LTFGRS/>

**BugReports** <https://github.com/EmilMiP/LTFGRS/issues>

**NeedsCompilation** yes

**Author** Emil Michael Pedersen [aut, cre],  
Jette Steinbach [aut],  
Lucas Rasmussen [ctb],  
Morten Dybdahl Krebs [aut]

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

**Repository** CRAN

**Date/Publication** 2025-08-28 22:40:11 UTC

## Contents

attach_attributes . . . . .	3
cancel_family_onsets . . . . .	4
construct_covmat . . . . .	5
construct_covmat_multi . . . . .	7
construct_covmat_single . . . . .	10
convert_age_to_cir . . . . .	11
convert_Age_to_thresh . . . . .	12
convert_cir_to_Age . . . . .	13
convert_format . . . . .	14
convert_liability_to_AoO . . . . .	15
convert_observed_to_liability_scale . . . . .	17
correct_positive_definite . . . . .	18
estimate_gen_liability_tfh . . . . .	20
estimate_liability . . . . .	21
estimate_liability_multi . . . . .	24
estimate_liability_single . . . . .	27
extract_estimation_info_graph . . . . .	29
extract_estimation_info_tbl . . . . .	30
familywise_attach_attributes . . . . .	30
familywise_censoring . . . . .	32
fixSexCoding . . . . .	33
get_all_combs . . . . .	33
get_covmat . . . . .	34
get_family_graphs . . . . .	35
get_onset_time . . . . .	36
get_relatedness . . . . .	37
Gibbs_estimator . . . . .	38
graph_based_covariance_construction . . . . .	39
graph_based_covariance_construction_multi . . . . .	40
graph_to_trio . . . . .	41
kendler . . . . .	43
kendler_family_calculations . . . . .	44
PA_algorithm . . . . .	45
prepare_graph . . . . .	46
prepare_thresholds . . . . .	47
rtmvnorm.gibbs . . . . .	49
simulate_under_LTM . . . . .	51
simulate_under_LTM_multi . . . . .	53
simulate_under_LTM_single . . . . .	55
tnorm_mean . . . . .	57
tnorm_mixture_conditional . . . . .	58

*attach\_attributes* 3

*tnorm\_var* . . . . . 58  
*truncated\_normal\_cdf* . . . . . 59

**Index** 61

---

*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,  
  fid,  
  attr_tbl,  
  attr_names,  
  censor_proband_thrs = TRUE  
)
```

### 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).

*fid*              Column name of family id.

*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.

*censor\_proband\_thrs*  
                  Should proband's upper and lower thresholds be made uninformative? Defaults to TRUE. Used to exclude proband's information for prediction.

### Value

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

---

censor\_family\_onsets *Censor onset times in a family based on a proband's end of follow-up.*

---

### Description

This function censors onset times for family members based on the proband's end of follow-up. This is done to prevent using future events to base predictions on.

### Usage

```
censor_family_onsets(
  tbl,
  proband_id_col,
  cur_proband,
  start,
  end,
  event,
  status_col = "status",
  aod_col = "aod",
  age_eof_col = "age"
)
```

### Arguments

tbl	tibble with info on family members, censoring events based on cur_proband in proband_id_col, must contain start, end, and event as columns
proband_id_col	column name of proband ids within family
cur_proband	current proband id
start	start of follow up, typically birth date, must be a date column
end	end of follow up, must be a date column
event	event of interest, typically date of diagnosis, must be a date column
status_col	column name of status column to be created. Defaults to "status".
aod_col	column name of age of diagnosis (aod) column to be created. Defaults to "aod".
age_eof_col	column name of age at end of follow-up (eof) column to be created. Defaults to "age_eof".

### Value

tibble with updated end times, status, age of diagnosis, and age at end of follow-up for a family, such that proband's end time is used as the end time for all family members. This prevents using future events to based predictions on.

### Examples

```
# See Vignettes.
```

---

construct_covmat	<i>Constructing a covariance matrix for a variable number of phenotypes</i>
------------------	---

---

## 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 <a href="#">setName</a> . 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

$$i \in \{1, \dots, n_{phenos}\}$$

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  $2 \times 2$  matrix holding only the correlation between the genetic component of the full liability and the full liability for the individual under consideration, or a

$$(2 \times n_{phenos}) \times (2 \times n_{phenos})$$

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 <a href="#">setNames</a> . 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

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

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

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

and

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

For two different phenotypes, the covariance is given by

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

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



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

and

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

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  $i \in \{1, \dots, n_{pheno}\}$ , 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 <a href="#">setNames</a> . 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  $2 \times 2$  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	<i>Convert age to cumulative incidence rate</i>
--------------------	---

---

### 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")
```

---

<code>convert_format</code>	<i>Attempts to convert the list entry input format to a long format</i>
-----------------------------	---

---

**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\_thresholds, which is used by estimate\_liability

**Examples**

```
family <- data.frame(  
  fid = 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

<code>obs_h2</code>	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
<code>pop_prev</code>	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 <code>obs_h2</code> . Defaults to 0.05.
<code>prop_cases</code>	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 <code>obs_h2</code> . 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](https://doi.org/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](https://doi.org/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 obs\_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\_cormat and full\_cormat 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	<i>Estimating the genetic or full liability for a variable number of phenotypes</i>
--------------------	---

---

**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",
  fid = "fid",
  role = "role",
  family_graphs_col = "fam_graph",
  out = c(1),
  tol = 0.01,
  method = "PA",
```

```

useMixture = FALSE,
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".
- fid** A string holding the name of the column in family that holds the family identifier. Defaults to "fid".
- 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.

method	Estimation method used to estimate the (genetic) liability. Defaults to "PA". Current implementation of PA only supports estimates of genetic liability. For full or both genetic and full liability estimates use "Gibbs".
useMixture	Logical indicating whether the mixture model should be used to calculate the genetic liability. Requires K_i and K_pop columns as well as lower and upper. Defaults to FALSE.
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 fid, 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 fid 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 fid 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", fid = "fid", 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",
  fid = "fid",
  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".
fid	A string holding the name of the column in family that holds the family identifier. Defaults to "fid".
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 `fid`, 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 `fid` 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", fid = "fid", 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",
  fid = "fid",
  family_graphs_col = "fam_graph",
  role = NULL,
  out = c(1),
  tol = 0.01,
  useMixture = FALSE,
  method = "PA"
)
```

## 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".

fid	A string holding the name of the column in .tbl or family that holds the family identifier. Defaults to "fid".
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.
useMixture	Logical indicating whether the mixture model should be used to calculate the genetic liability. Requires K_i and K_pop columns as well as lower and upper. Defaults to FALSE.
method	Estimation method used to estimate the (genetic) liability. Defaults to "PA". Current implementation of PA only supports estimates of genetic liability. For full or both genetic and full liability estimates use "Gibbs".

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

---

extract\_estimation\_info\_graph

*Title Internal Function used to extract input needed from graph input for liability estimation*

---

**Description**

Title Internal Function used to extract input needed from graph input for liability estimation

**Usage**

```
extract_estimation_info_graph(cur_fam_graph, cur_fid, h2, pid, add_ind = TRUE)
```

**Arguments**

cur_fam_graph	neighbourhood graph of degree n around proband
cur_fid	proband ID
h2	heritability value from estimate_liability
pid	Name of column of personal ID
add_ind	Whether the genetic liability be added. Default is TRUE.

**Value**

list with two elements: tbl (tibble with all relevant information) and cov (covariance matrix) estimated through graph\_based\_covariance\_construction()

---

extract\_estimation\_info\_tbl

*Title Internal Function used to extract input needed for liability estimation*

---

### Description

Title Internal Function used to extract input needed for liability estimation

### Usage

```
extract_estimation_info_tbl(.tbl, cur_fid, h2, fid, pid, role, add_ind = TRUE)
```

### Arguments

.tbl	.tbl input from estimate_liability
cur_fid	current family ID being worked on
h2	heritability value from estimate_liability
fid	name of family ID column
pid	name of personal ID column
role	name of role column
add_ind	Whether the genetic liability be added. Default is TRUE.

### Value

list with two elements: tbl (tibble with all relevant information) and cov (covariance matrix) estimated through construct\_covmat()

---

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"),  
  censor_proband_thrs = TRUE  
)
```

**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. defaults 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.

**censor\_proband\_thrs** 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.
```

---

familywise\_censoring *Censor Family Onsets for Multiple Families*

---

### Description

This function is a wrapper around `censor_family_onsets`. This function accepts a tibble with family graphs from `get_family_graphs`. It censors the onset times for each individual in the family graph based on the proband's end of follow-up. Returns a formatted output.

### Usage

```
familywise_censoring(
  family_graphs,
  tbl,
  start,
  end,
  event,
  status_col = "status",
  aod_col = "aod",
  age_eof_col = "age",
  fam_graph_col = "fam_graph",
  fid = "fid",
  pid = "pid",
  merge_by = pid
)
```

### Arguments

<code>family_graphs</code>	Tibble with <code>fid</code> and family graphs columns.
<code>tbl</code>	Tibble with information on each considered individual.
<code>start</code>	Column name of start of follow up, typically date of birth.
<code>end</code>	Column name of the personalised end of follow up.
<code>event</code>	Column name of the event.
<code>status_col</code>	Column name of the status (to be created). Defaults to "status".
<code>aod_col</code>	Column name of the age of diagnosis (to be created). Defaults to "aod".
<code>age_eof_col</code>	Column name of the age at the end of follow up (to be created). Defaults to "age_eof".
<code>fam_graph_col</code>	Column name of family graphs in the 'family_graphs' object. Defaults to "fam_graph".
<code>fid</code>	Family id, typically the name of the proband that a family graph is centred on. Defaults to "fid".
<code>pid</code>	Personal identifier for each individual. Allows for multiple instances of the same individual across families. Defaults to "pid".
<code>merge_by</code>	Column names to merge by. If different names are used for family graphs and <code>tbl</code> , a named vector can be specified: <code>setNames(c("id"), c("pid"))</code> . Note <code>id</code> is the column name in <code>tbl</code> and <code>pid</code> is the column name in <code>family_graphs</code> . The column names used should reference the personal identifier.



**Value**

A tibble with family ids and updated status, age of diagnosis, and age at end of follow-up for each individual in the family based on the proband's end of follow-up.

**Examples**

```
# See Vignettes.
```

---

fixSexCoding	<i>Fixing sex coding in trio info</i>
--------------	---------------------------------------

---

**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	<i>construct all combinations of input vector</i>
---------------	---

---

**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_covmat
```

*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_covmat(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 conjunction 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_covmat(graph, h2 = 0.5, index_id = "1")
```

---

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

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

### Value

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

### Examples

```
# See Vignettes.
```

---

get_onset_time	<i>Calculate age of diagnosis, age at end of follow up, and status</i>
----------------	--

---

### Description

Calculate age of diagnosis, age at end of follow up, and status

### Usage

```
get_onset_time(  
  tbl,  
  start,  
  end,  
  event,  
  status_col = "status",  
  aod_col = "aod",  
  age_eof_col = "age"  
)
```

### Arguments

tbl	tibble with start, end, and event as columns
start	start of follow up, typically birth date, must be a date column
end	end of follow up, must be a date column
event	event of interest, typically date of diagnosis, must be a date column
status_col	column name of status column to be created. Defaults to "status".
aod_col	column name of age of diagnosis column to be created. Defaults to "aod".
age_eof_col	column name of age at end of follow-up column to be created. Defaults to "age_eof".

### Value

tibble with added status, age of diagnosis, and age at end of follow-up

### Examples

```
# See vignettes.
```

---

get_relatedness	<i>Relatedness between a pair of family members</i>
-----------------	---

---

**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))
```

---

Gibbs_estimator	<i>Wrapper around the Gibbs Sampler that returns formatted liability estimates for the proband</i>
-----------------	--

---

### Description

Wrapper around the Gibbs Sampler that returns formatted liability estimates for the proband

### Usage

```
Gibbs_estimator(cov, tbl, out, tol = 0.01, burn_in = 1000)
```

### Arguments

cov	Covariance (kinship matrix times heritability with corrected diagonal) matrix
tbl	Tibble with lower and upper bounds for the Gibbs sampler
out	Vector indicating if genetic ans/or full liabilities should be estimated
tol	Convergence criteria, tolerance
burn_in	Number of burn-in iterations

### Value

Formatted liability estimate(s) and standard error(s) of the mean for the proband.

### Examples

```
# uninformative sampling:
Gibbs_estimator(cov = diag(3), tbl = tibble::tibble(lower = rep(-Inf, 3),
upper = rep(Inf, 3)), out = 1:2, tol = 0.01, burn_in = 1000)
```

---

`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

<code>pid</code>	Name of column of personal ID
<code>cur_proband_id</code>	id of proband
<code>cur_family_graph</code>	local graph of current proband
<code>h2</code>	liability scale heritability
<code>add_ind</code>	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(
  fid,
  pid,
  cur_proband_id,
  cur_family_graph,
  h2_vec,
  genetic_corrmat,
  phen_names,
  add_ind = TRUE
)

```

### Arguments

fid	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(fid = "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(),
  lower = sample(c(-Inf, 2), size = length(id), replace = TRUE),
  upper = sample(c(2, Inf), size = length(id), replace = TRUE),
  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)
}

```

---

kendler

*Title Kendler's FGRS*


---

## Description

Title Kendler's FGRS

## Usage

```

kendler(
  .tbl = NULL,
  family_graphs = NULL,
  family_graphs_col = "fam_graph",
  pid = "pid",
  fid = "fid",
  role = NULL,
  dadcol,
  momcol,
  env_cor_sib = 1,
  env_cor_f = 1,
  env_cor_m = 1
)

```

## 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.
family_graphs_col	Name of column with family graphs in family_graphs. Defaults to "fam_graph".
pid	A string holding the name of the column in .tbl (or family and threshs) that hold the personal identifier(s). Defaults to "PID".
fid	A string holding the name of the column in .tbl or family that holds the family identifier. Defaults to "fid".
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".
dadcol	column name of father in family_graphs or .tbl.
momcol	column name of mother in family_graphs or .tbl.
env_cor_sib	Cohabitation effect, i.e. Factor by which the siblings are weighted. Defaults to 1.
env_cor_f	Cohabitation effect, i.e. Factor by which the father is weighted. Defaults to 1.
env_cor_m	Cohabitation effect, i.e. Factor by which the mother is weighted. Defaults to 1.

### Value

A tibble with summary values used to calculate the kendler FGRS and the FGRS itself.

### Examples

```
# See Vignettes.
```

---

```
kendler_family_calculations
```

*Title Helper function for Kendler's FGRS*

---

### Description

Title Helper function for Kendler's FGRS

**Usage**

```
kendler_family_calculations(
  tbl,
  cov,
  pid,
  cur_dad_id,
  cur_mom_id,
  env_cor_sib = 1,
  env_cor_f = 1,
  env_cor_m = 1
)
```

**Arguments**

tbl	tibble with columns cip, lower, upper, and pid (the personal identifier column).
cov	Kinship matrix with proband as first row and column
pid	column name of personal identifier
cur_dad_id	ID of father (not column name, but the actual ID)
cur_mom_id	ID of mother (not column name, but the actual ID)
env_cor_sib	Cohabitation effect, i.e. Factor by which the siblings are weighted. Defaults to 1.
env_cor_f	Cohabitation effect, i.e. Factor by which the father is weighted. Defaults to 1.
env_cor_m	Cohabitation effect, i.e. Factor by which the mother is weighted. Defaults to 1.

**Value**

A tibble with family specific values required for Kendler's FGRS calculation.

**Examples**

```
# See Vignettes.
```

---

PA_algorithm	<i>Title Pearson-Aitken algorithm to calculate mean values in truncated multivariate normal distributions</i>
--------------	---

---

**Description**

Title Pearson-Aitken algorithm to calculate mean values in truncated multivariate normal distributions

**Usage**

```
PA_algorithm(mu, covmat, target_id, lower, upper, K_i = NA, K_pop = NA)
```

**Arguments**

mu	vector of means
covmat	covariance matrix, containing kinship coefficient and heritability on each entry (except diagonal, which is 1 for full liabilities and h2 for genetic liabilities)
target_id	ID of target individual (or genetic liability), i.e. rowname in covmat to return expected genetic liability for
lower	vector of lower thresholds
upper	vector of upper thresholds
K_i	vector of stratified CIPs for each individual. Only used for estimating genetic liability under the mixture model.
K_pop	vector of population CIPs. Only used for estimating genetic liability under the mixture model.

**Value**

A list with two elements: est (expected genetic liability, given input data) and var (variance of genetic liability, given input data).

---

prepare_graph	<i>Construct graph from register information</i>
---------------	--

---

**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\_thresholds      *Calculate (personalised) thresholds based on CIPs.*

---

**Description**

This function prepares input for `estimate_liability` by calculating thresholds based on stratified cumulative incidence proportions (CIPs) with options for interpolation for ages between CIP values. Given a tibble with families and family members and (stratified) CIPs, personalised thresholds will be calculated for each individual present in .tbl. An individual may be in multiple families, but only once in the same family.

**Usage**

```
prepare_thresholds(
  .tbl,
  CIP,
  age_col,
  CIP_merge_columns = c("sex", "birth_year", "age"),
  CIP_cip_col = "cip",
```

```

Kpop = "useMax",
status_col = "status",
lower_equal_upper = FALSE,
personal_thr = FALSE,
fid_col = "fid",
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 = 30
)

```

### Arguments

<code>.tbl</code>	Tibble with family and personal id columns, as well as <code>CIP_merge_columns</code> and <code>status</code> .
<code>CIP</code>	Tibble with population representative cumulative incidence proportions. <code>CIP</code> must contain columns from <code>CIP_merge_columns</code> and <code>cIP_cip_col</code> .
<code>age_col</code>	Name of column with age at the end of follow-up or age at diagnosis for cases.
<code>CIP_merge_columns</code>	The columns the CIPs are subset by, e.g. CIPs by <code>birth_year</code> , <code>sex</code> .
<code>CIP_cip_col</code>	Name of column with CIP values.
<code>Kpop</code>	Takes either "useMax" to use the maximum value in the CIP strata as population prevalence, or a tibble with population prevalence values based on other information. If a tibble is provided, it must contain columns from <code>.tbl</code> and a column named "K_pop" with population prevalence values. Defaults to "UseMax".
<code>status_col</code>	Column that contains the status of each family member. Coded as 0 or FALSE (control) and 1 or TRUE (case).
<code>lower_equal_upper</code>	Should the upper and lower threshold be the same for cases? Can be used if CIPs are detailed, e.g. stratified by birth year and sex.
<code>personal_thr</code>	Should thresholds be based on stratified CIPs or population prevalence?
<code>fid_col</code>	Column that contains the family ID.
<code>personal_id_col</code>	Column that contains the personal ID.
<code>interpolation</code>	Type of interpolation, defaults to NULL.
<code>bst.params</code>	List of parameters to pass on to xgboost. See xgboost documentation for details.
<code>min_CIP_value</code>	Minimum cip value to allow. Too low values may lead to numerical instabilities.
<code>xgboost_itr</code>	Number of iterations to run xgboost for.

### Value

Tibble with (personalised) thresholds for each family member (lower & upper), the calculated cumulative incidence proportion for each individual ( $K_i$ ), and population prevalence within an individuals CIP strata ( $K_{pop}$ ; max value in stratum). The threshold and other potentially relevant information can be added to the family graphs with `familywise_attach_attributes`.



**Examples**

```
tbl = data.frame(
  fid = 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_thresholds(.tbl = tbl, CIP = cip, age_col = "age", interpolation = NA)
```

---

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 $-\text{Inf}$ .
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 $\text{Inf}$ .
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 <code>out = c(1)</code> , the first variable (usually the genetic component of the full liability of the first phenotype) is estimated and returned. If <code>out = c(2)</code> , the second variable (usually full liability) is estimated and returned. If <code>out = c(1, 2)</code> , both the first and the second variable are estimated and returned. Defaults to <code>c(1)</code> .
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 <a href="#">setName</a> . 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 $\text{length}(h2) > 0$ , and will be ignored if $h2$ is a number. All diagonal entries in <code>full_corrmat</code> 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 $\text{length}(h2) > 0$ , and will be ignored if $h2$ is a number. If it is not specified, the names will default to <code>phenotype1</code> , <code>phenotype2</code> , 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 <code>pop_prev</code> 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  $i \in \{1, \dots, n_{phenos}\}$ , `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 <a href="#">setNames</a> . 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  $i \in \{1, \dots, n_{pheno}\}$ , `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 <a href="#">setName</a> . 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  $h^2$  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)
```

---

tnorm\_mean

*Title: Calculate the mean of the truncated normal distribution*


---

**Description**

Title: Calculate the mean of the truncated normal distribution

**Usage**

```
tnorm_mean(mu = 0, sigma = 1, lower = -Inf, upper = Inf)
```

**Arguments**

mu	mean value of normal distribution
sigma	standard deviation of normal distribution
lower	lower threshold
upper	upper threshold

**Value**

mean value of the truncated normal distribution

**Examples**

```
tnorm_mean()
```

---

 tnorm\_mixture\_conditional

*Title: Calculates mean and variance of mixture of two truncated normal distributions*

---

### Description

Title: Calculates mean and variance of mixture of two truncated normal distributions

### Usage

```
tnorm_mixture_conditional(mu, var, lower, upper, K_i, K_pop)
```

### Arguments

mu	Mean value of normal distribution.
var	Variance of normal distribution.
lower	Lower threshold (can be -Inf).
upper	Upper threshold (can be Inf).
K_i	(Stratified) cumulative incidence proportion for the individual.
K_pop	Population prevalence (cumulative incidence proportion).

### Value

mean and variance of mixture distribution between two truncated normal distributions

### Examples

```
tnorm_mixture_conditional(mu = 0, var = 1, lower = -Inf, upper = Inf, K_i = 0, K_pop = 0.01)
tnorm_mixture_conditional(mu = 0, var = 1, lower = -Inf, upper = 2, K_i = .01, K_pop = 0.05)
```

---

 tnorm\_var

*Title: Calculate the variance of the truncated normal distribution*

---

### Description

Title: Calculate the variance of the truncated normal distribution

### Usage

```
tnorm_var(mu = 0, sigma = 1, lower = -Inf, upper = Inf)
```

**Arguments**

mu	mean value of normal distribution
sigma	standard deviation of normal distribution
lower	lower threshold
upper	upper threshold

**Value**

mean value of the truncated normal distribution

**Examples**

```
tnorm_var()
```

---

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

attach\_attributes, 3

    censor\_family\_onsets, 4

    construct\_covmat, 5, 9, 11, 19, 53, 55, 57

    construct\_covmat\_multi, 5, 7, 7, 11, 19

    construct\_covmat\_single, 5, 7, 9, 10, 19

    convert\_age\_to\_cir, 11

    convert\_age\_to\_thresh, 12

    convert\_cir\_to\_age, 13

    convert\_format, 14

    convert\_liability\_to\_aoo, 15

    convert\_observed\_to\_liability\_scale, 17

    correct\_positive\_definite, 18

    estimate\_gen\_liability\_ltfh, 20

    estimate\_liability, 21, 26, 29, 52, 53, 55, 56

    estimate\_liability\_multi, 21, 24, 24, 29

    estimate\_liability\_single, 21, 24, 26, 27

    extract\_estimation\_info\_graph, 29

    extract\_estimation\_info\_tbl, 30

    familywise\_attach\_attributes, 30

    familywise\_censoring, 32

    fixSexCoding, 33

    future\_apply, 24, 26, 29

    get\_all\_combs, 33

    get\_covmat, 34

    get\_family\_graphs, 35

    get\_onset\_time, 36

    get\_relatedness, 7, 9, 11, 37

    Gibbs\_estimator, 38

    graph\_based\_covariance\_construction, 39

    graph\_based\_covariance\_construction\_multi, 40

    graph\_to\_trio, 41

kendler, 43

kendler\_family\_calculations, 44

PA\_algorithm, 45

prepare\_graph, 46

prepare\_thresholds, 47

rtmvnorm.gibbs, 49

setNames, 5, 8, 10, 51, 54, 56

simulate\_under\_LTM, 51, 57

simulate\_under\_LTM\_multi, 53, 53, 57

simulate\_under\_LTM\_single, 53, 55

tnorm\_mean, 57

tnorm\_mixture\_conditional, 58

tnorm\_var, 58

truncated\_normal\_cdf, 59