Package: GWASBrewer (via r-universe)

December 12, 2024

Type Package

Title Simulate Realistic GWAS Summary Statistics

Version 0.3.0.0214

Author Jean Morrison

Maintainer Jean Morrison <jvmorr@umich.edu>

Description Simulate GWAS summary statistics from specified DAG or factor structure.

License GPL (>= 3)

URL https://github.com/jean997/GWASBrewer

BugReports https://github.com/jean997/GWASBrewer/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Depends R (>= 4.1.0)

Imports dplyr, MASS, Matrix, purrr, reshape2, stringr, tidyr

VignetteBuilder knitr

Suggests DiagrammeR, knitr, ggplot2, rmarkdown, hapsim, testthat (>= 3.0.0)

Config/testthat/edition 3

LazyDataCompression xz

Config/pak/sysreqs libicu-dev

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

RemoteUrl https://github.com/jean997/GWASBrewer

RemoteRef HEAD

RemoteSha fdea10e7c4d86585a42c918a2ab828103e07b5ae

Contents

`compute_h2` *Compute heritability from standardized or non-standardized effects*

Description

Compute heritability from standardized or non-standardized effects

Usage

```
compute_h2(
  b_joint,
  geno_scale = c("allele", "sd"),
  pheno_sd = 1,
  R_LD = NULL,
  af = NULL,
  full_mat = FALSE
)
```

Arguments

`R_LD`	LD pattern (optional). See `?sim_mv` for more details.
`af`	Allele frequencies (optional, allowed only if `R_LD` is missing). See `?sim_mv` for more details.
`full_mat`	If TRUE, return the full genetic variance-covariance matrix
`b_joint_std, b_joint`	matrix of standardized or non-standardized effects. Provide only one of these options.

`generate_random_F` *Generate random F*

Description

Generate random F

Usage

```
generate_random_F(
  K,
  M,
  g_F = function(n) {
     stats::runif(n, -1, 1)
 },
  nz_factor,
  omega,
  h2_trait,
  pad = FALSE
)
```

Arguments

`g_F`	Function from which non-zero elements of F are generated
`nz_factor`	Number of non-zero elements of each factor if F is to be generated.
`omega`	Proportion of trait heritability explained by factors
`h2_trait`	Trait heritability
`pad`	Add single trait factors? (See details)

Details

Generate a random set of (at least) K factors for M traits. The number of traits affected by each factor is given by `nz_factor`. Each effect is chosen as a random draw from function g_F. If any rows of the resulting matrix corresponding to non-zero elements of omega are all zero and pad = TRUE, single-trait factors are added. Finally, the matrix is re-scaled so that `colSums(F_mat^2) = omega*h2_trait`.

Value

A matrix

gen_bhat_from_b *Generate beta hats from standardized or non-standardized direct SNP effects and LD*

Description

Generate beta hats from standardized or non-standardized direct SNP effects and LD

Usage

```
gen_bhat_from_b(
  b_joint,
  N,
  trait_corr = NULL,
  R_LD = NULL,
  af = NULL,
  est_s = FALSE,
  input_geno_scale = c("allele", "sd"),
  input_pheno_sd = 1,
  output_geno_scale = c("allele", "sd"),
  output_pheno_sd = 1
)
```

Arguments

b_joint	Matrix of joint effect sizes
N	Sample size, scalar, vector, matrix, or data.frame. See ?sim_mv for more details.
trait_corr	Matrix of population trait correlation (traits by traits)
R_LD	LD pattern (optional). See ?sim_mv for more details.
af	Allele frequencies (optional, allowed only if R_LD is missing). See ?sim_mv for more details.
est_s	Estimate standard errors?
input_geno_scale	Genotype scale of effects in b_joint
input_pheno_sd	Phenotype sd for effects in b_joint
output_geno_scale	Output genotype scale
output_pheno_sd	Output phenotype sd

Details

This has been made an internal function. To resample summary statistics use resample_sumstats.

GWASBrewer *GWASBrewer*

Description

Simulate GWAS summary statistics from specified DAG or factor structure.

Author(s)

Jean Morrison <jvmorr@umich.edu>

See Also

Useful links:

- https://github.com/jean997/GWASBrewer
- Report bugs at https://github.com/jean997/GWASBrewer/issues

resample_inddata *Sample individual level data with joint effects matching a sim_mv object*

Description

Sample individual level data with joint effects matching a sim_mv object

Usage

```
resample_inddata(
  N,
  dat = NULL,
  genos = NULL,
  J = NULL,
  R_LD = NULL,
  af = NULL,
  sim_func = gen_genos_mvn,
  new_env_var = NULL,
  new_h2 = NULL,
  new_R_E = NULL,
  new_R_obs = NULL,
  calc_sumstats = FALSE
)
```

Arguments

N	Sample size, scalar, vector, or special sample size format data frame, see details.
dat	An object of class sim_mv (produced by sim_mv). If 'dat' is omitted, the function will generate a matrix of genotypes only. If 'dat' is provided, phenotypes for the traits in 'dat' will also be included.
genos	Optional matrix of pre-generated genotypes. If genos is supplied, resample_inddata will only generate phenotypes.
J	Optional number of variants. J is only required if dat is missing.
R_LD	LD pattern (optional). See ?sim_mv for more details.
af	Allele frequencies. af is required unless unless genos is supplied.
new_env_var	Optional. The environmental variance in the new population. If missing the function will assume the environmental variance is the same as in the old population.
new_h2	Optional. The heritability in the new population. Provide at most one of new_env_var and new_h2.
new_R_E	Optional, specify environmental correlation in the new population. If missing, the function will assume the environmental correlation is the same as in the original data.
new_R_obs	Optional, specify overall trait correlation in the new population. Specify at most one of new_R_E or new_R_obs. If missing, the function will assume the environmental correlation is the same as in the original data.
calc_sumstats	If TRUE, associations between genotypes and phenotypes will be calculated and returned.

Details

This function can be used to generate individual level genotype and phenotype data. It can be used in three modes:

To generate genotype data only: No sim_mv object needs to be included. Supply only N as a single integer for the number of individuals, J for the number of variants, af, and R_LD if desired. All other parameters are not relevant if there is no phenotype, so if they are supplied, you will get an error. The returned object will include a N x J matrix of genotypes and a vector of allele frequencies.

To generate both genotype and phenotype data: Supply dat (a sim_mv object) and leave genos missing. N and af are required and all other options are optional.

To generate phenotype data only: Supply dat (a sim_mv object) and provide a matrix of genotypes to the genos argument. The number of rows in genos must be equal to the total number of individuals implied by N. So for example, if there are two traits with 10 samples each and no overlap, genos should have 20 rows. The R_LD and af arguments should contain the population LD and allele frequencies used to produce the genotypes. These are used to compute the genetic variance-covariance matrix. N and af are required and all other options are optional.

Examples

```
# Use resample_inddata to generate genotypes only
simple_ld <- matrix(0.5, nrow = 5, ncol = 5)
diag(simple_ld) <- 1
genos_only <- resample_inddata(N = 8,
                               J = 20,
                               R_LD = list(simple_ld),
                               af = rep(0.3, 5))
# generate genotypes and phenotypes
dat <- sim_mv(N = 0,
              G = 1,
              J = 20,
              pi = 0.5,
              h2 = 0.05,
              R_LD = list(simple_ld),
              af = rep(0.3, 5))
genos_and_phenos <- resample_inddata(dat = dat,
                                     N = 8,
                                     R_LD = list(simple_ld),
                                     af = rep(0.3, 5))
# generate phenos only
phenos_only <- resample_inddata(dat = dat,
                                genos = genos_only$X,
                                N = 8,
                                R_LD = list(simple_ld),
                                af = rep(0.3, 5))
```

`resample_sumstats` *Resample Summary Statistics for Existing Simulation Object*

Description

Resample Summary Statistics for Existing Simulation Object

Usage

```
resample_sumstats(
  dat,
  N,
  R_LD = NULL,
  af = NULL,
  est_s = FALSE,
  geno_scale = NULL,
  new_env_var = NULL,
  new_h2 = NULL,
  new_R_E = NULL,
  new_R_obs = NULL
)
```

Arguments

dat	Object output by sim_mv
N	Sample size, scalar, vector, matrix. See ?sim_mv for more details.
R_LD	LD pattern (optional). See ?sim_mv for more details.
af	Allele frequencies. See ?sim_mv for more details.
est_s	Logical, should estimates of se(beta_hat) be produced.
geno_scale	Either "allele" or "sd". Specifies the scale of the effect sizes in the output data.
new_env_var	Optional. The environmental variance in the new population. If missing the function will assume the environmental variance is the same as in the old population.
new_h2	Optional. The heritability in the new population. Provide at most one of new_env_var and new_h2.
new_R_E	Optional, specify environmental correlation in the new population. If missing, the function will assume the environmental correlation is the same as in the original data.
new_R_obs	Optional, specify overall trait correlation in the new population. Specify at most one of new_R_E or new_R_obs. If missing, the function will assume the environmental correlation is the same as in the original data.

Details

This function can be used to generate new summary statistics for an existing simulation object. For a discussion of this function and resample_inddata, see the "Resampling" vignette.

Examples

```
# Use resample_sumstats to generate new GWAS results with the same effect sizes.
N <- matrix(1000, nrow = 2, ncol =2)
G <- matrix(0, nrow = 2, ncol = 2)
R_E <- matrix(c(1, 0.8, 0.8, 1), nrow = 2, ncol = 2)
# original data
dat <- sim_mv(N = N, J = 20000, h2 = c(0.4, 0.3), pi = 1000/20000,
              G = G, R_E = R_E)
# data for second GWAS
dat_new <- resample_sumstats(dat,
                             N = 40000)
```

---

rescale_sumstats *Re-Scale Effects of a Simulation Object*

Description

Re-Scale Effects of a Simulation Object

Usage

```
rescale_sumstats(
  dat,
  output_geno_scale = c("allele", "sd"),
  output_pheno_sd = 1,
  af = NULL,
  verbose = TRUE
)
```

Arguments

dat	A sim_mv object
output_geno_scale	Desired genotype scale of output. Either "allele" or "sd".
output_pheno_sd	Desired sd of phenotype, scalar or vector with length equal to the number of traits.
af	If converting from sd to allele scale, provide a vector of allele frequencies.
verbose	Print messages?

Details

This function can be used to change the genotype and phenotype scaling. To check the scaling of the current object, look at the gneo_scale and pheno_sd elements. If the current object is already on the allele scale and you desire the output to also be on the allele scale, do not supply af (doing so will generate an error). If you convert an "allele" scale object to an "sd" scale object, allele frequencies will be remoed.

Examples

```
# generate an initial data set
N <- matrix(10000, nrow = 2, ncol =2)
G <- matrix(c(0, 0.5, 0, 0), nrow = 2, ncol = 2)
dat <- sim_mv(N = N,
              G = G,
              J = 20000,
              h2 = c(0.4, 0.3),
              pi = 1000/20000,
              af = function(n){rbeta(n, 1, 5)})
# check scaling
dat$geno_scale # "allele"
dat$pheno_sd # 1 1

# rescale phenotypes and convert to per-sd scale
dat2 <- rescale_sumstats(dat = dat,
                         output_geno_scale = "sd",
                         output_pheno_sd = c(1.5, 0.3))
```

rnormalmix *Simulate from a normal mixture distribution*

Description

Simulate from a normal mixture distribution

Usage

```
rnormalmix(n, sd, pi, mu = 0, return.Z = FALSE)
```

Arguments

n	Number of points to simulate
pi	Mixture proportions
mu	Means
return.Z	if TRUE, also return a vector of indicators indicating which of the K classes each sample belongs to
sigma	Standard deviations

Value

If return.Z=TRUE, returns a list with elements beta (samples) and Z (indicators). Otherwise returns a length n vector of samples.

sim_extract_ld *Extract LD matrix from simulated data*

Description

Extract LD matrix for specific variants in simulated data set

Usage

```
sim_extract_ld(dat, index, R_LD)
```

Arguments

dat	Simulation object produced by ‘sim_mv‘
index	vector of indices for snps to extract LD
R_LD	List of eigen-decompositions used in original simulation

Value

An LD matrix. SNP order matches original index order

Examples

```
data("ld_mat_list")
data("AF")
# Two traits with no causal relationship, non-overlapping GWAS
# Set N = 0 so no summary statistics are produced
set.seed(1)
G <- matrix(0, nrow = 2, ncol = 2)
dat <- sim_mv(N = 0, J = 1000, h2 = c(0.04, 0.03), pi = 0.1,
              G = G,  R_LD = ld_mat_list, af = AF)

# extract ld matrix for all variants with p-value for trait 1 less than 1e-5
index <- c(1:20, 500:515)
ld_mat <- sim_extract_ld(dat, index, ld_mat_list)
```

`sim_ld_proxy` *Retrieve LD proxies*

Description

Retrieve LD proxies

Usage

```
sim_ld_proxy(dat, index, R_LD, r2_thresh = 0.64, return_mat = FALSE)
```

Arguments

`dat`	Simulation object produced by 'sim_mv'
`index`	list of indexes to retrieve proxies for
`R_LD`	LD pattern used to generate dat
`r2_thresh`	Get proxies with r^2 >= r2_thresh with one of the index variants
`return_mat`	If TRUE, return the correlation matrix between the index variant and the proxies. In this matrix, the the index variant always corresponds the first row/column and the proxies are in the order returned.

`sim_ld_prune` *LD prune simulated data*

Description

LD prune simulated data

Usage

```
sim_ld_prune(dat, pvalue, R_LD, r2_thresh = 0.1, pval_thresh = 1)
```

Arguments

dat	Data object produced by sim_mv
pvalue	Either a vector used to prioritize variants or an integer. If pvalue is an integer, i, variants will be priorized by the p-value for trait i. If pvalue is missing, variants will be prioritized randomly.
R_LD	LD pattern used to generate dat
r2_thresh	r^2 threshold for pruning
pval_thresh	p-value threshold for pruning (see details)

Details

Given results from sim_mv, and a vector of p-values, the function will return a list of variants that have p < pval_thresh and which mutually have squared correlation less than r2_thresh.

Value

A vector of indices corresponding to the LD-pruned variant set.

Examples

```
data("ld_mat_list")
data("AF")

# Two traits with no causal relationship, non-overlapping GWAS
set.seed(1)
G <- matrix(0, nrow = 2, ncol = 2)
dat <- sim_mv(N = 10000, J = 1000, h2 = c(0.04, 0.03), pi = 0.1,
              G = G,  R_LD = ld_mat_list, af = AF)

# prune on p-value for first trait
pvals <- 2*pnorm(-abs(dat$beta_hat/dat$se_beta_hat))
prune_set_1 <- sim_ld_prune(dat, pvalue = pvals[,1], R_LD = ld_mat_list, pval_thresh = 1e-5)
# Above is equivalent to
prune_set_1 <- sim_ld_prune(dat, pvalue = 1, R_LD = ld_mat_list, pval_thresh = 1e-5)
```

---

sim_lf *Simulate summary statistics*

Description

Simulate summary statistics from a specified factor structure

Usage

```
sim_lf(
  F_mat,
  N,
  J,
  h2_trait,
  omega,
  h2_factor,
  pi_L,
  pi_theta,
  est_s = FALSE,
  R_E = NULL,
  R_obs = NULL,
  R_LD = NULL,
  af = NULL,
  sporadic_pleiotropy = TRUE,
  h2_exact = FALSE,
  pi_exact = FALSE,
  snp_effect_function_L = "normal",
  snp_effect_function_theta = "normal",
  snp_info = NULL
)
```

Arguments

F_mat	factor matrix M by K (M = number of traits, K = number of factors)
N	GWAS sample size. N can be a scalar, vector, or matrix. If N is a scalar, all GWAS have the same sample size and there is no overlap between studies. If N is a vector, each element of N specifies the sample size of the corresponding GWAS and there is no overlap between studies. If N is a matrix, N_ii specifies the sample size of study i and N_ij specifies the number of samples present in both study i and study j. The elements of N must be positive but non-integer values will not generate an error.
J	Total number of SNPs to generate
h2_trait	Heritability of each trait. Length M vector.
omega	Proportion of trait heritability mediated by factors. Length M vector.
h2_factor	Heritability of each factor. Length K vector.
pi_L	Proportion of non-zero elements in L_k. Length K factor
pi_theta	Proportion of non-zero elements in theta. Scalar or length M vector.
est_s	If TRUE, return estimates of se(beta_hat).
R_E	Correlation between environmental trait components not mediated by factors. M by M pd matrix.
R_obs	Observational correlation between traits. M by M pd matrix. At most one of R_E and R_obs can be specified.

R_LD — List of LD blocks. R_LD should have class `list`. Each element of R_LD can be either a) a matrix, b) a sparse matrix (class `dsCMatrix`) or c) an eigen decomposition (class `eigen`). All elements should be correlation matrices, meaning that they have 1 on the diagonal and are positive definite.

af — Optional vector of allele frequencies. If R_LD is not supplied, af can be a scalar, vector or function. If af is a function it should take a single argument (n) and return a vector of n allele frequencies (See Examples). If R_LD is supplied, af must be a vector with length equal to the size of the supplied LD pattern (See Examples).

sporadic_pleiotropy — Allow sporadic pleiotropy between traits. Defaults to TRUE.

h2_exact — If TRUE, the heritability of each trait will be exactly h2.

pi_exact — If TRUE, the number of direct effect SNPs for each trait will be exactly equal to round(pi*J).

snp_effect_function_L, snp_effect_function_theta — Optional function to generate variant effects in `L` or `theta`. `snp_effect_function_L/theta` can be a single function or list of functions of length equal to the number of factors/number of traits.

snp_info — Optional `data.frame` of variant information to be passed to variant effect functions. If `R_LD` is specified, `snp_info` should have number of rows equal to the size of the supplied LD pattern. Otherwise `snp_info` should have `J` rows.

Details

This function will generate GWAS summary statistics for M traits with K common factors. The matrix F_mat provides the effects of each factor on each trait, `F_mat[i,j]` gives the effect of factor j on trait i. The rows of `F_mat` will be scaled in order to provide desired proportion of heritability of each trait explained by factors but the relative size and sign of elements within rows will be retained.

A random factor matrix can be generated using `generate_random_F` (see Examples).

It is possible to supply a non-feasible set of parameters. Usually this occurs if the heritability of the factors is low but the heritability of the traits is high leading to a contradiction. The function will return an error if this happens.

Trait covariance: Each trait is composed of four independent components, the genetic component mediated by factors, the environmental component mediated by factors, the genetic component not mediated by factors, and the environmental component not mediated by factors. Therefore, the total trait covariance can be decomposed into the sum of four corresponding covariance matrices.

$$Cov(T) = Sigma_{FG} + Sigma_{FE} + Sigma_{GDir} + Sigma_{EDir}$$

We assume that all cross-trait genetic sharing is explained by the factors so that $Sigma_{GDir}$ is diagonal. Each factor is a sum of a genetic component and an environmental components and factors are independent (both genetic and environmental components) are independent across factors. This means that $Sigma_{FG} = FS_{FG}F^T$ and $Sigma_{FE} = FS_{FE}F^T$ where S_{FG} and S_{FE} are diagonal matrices. The parameter `R_E` specifies the correlation of the residual environmental component (i.e. $R_E = cov2cor(Sigma_{EDir})$. Alternatively, if `R_obs` is specified, $Sigma_{EDir}$ will be chosen to give the desired observational correlation. In the returned object, `Sigma_G` is equal to the sum of

the two genetic covariance components and `Sigma_E` is equal to the sum of the two environmental components. `R` gives the overall trait correlation matrix multiplied by the overlap proportion matrix, which is equal to the correlation in the error terms of `beta_hat` (See Examples).

Examples

```
myF <- generate_random_F(K = 3, M = 10, nz_factor = c(2, 3, 2),
                         omega = rep(0.8, 10),
                         h2_trait = rep(0.6, 10), pad = TRUE)
dat <- sim_lf(myF, N = 10000, J = 20000, h2_trait = rep(0.6, 10),
                       omega = rep(0.8, 10), pi_L = 0.1, pi_theta = 0.1)

myF <- diag(2)
N <- matrix(c(10000, 8000, 8000, 10000), nrow = 2)
R_E <- matrix(c(1, 0.6, 0.6, 1), nrow = 2)
dat <- sim_lf(F_mat = myF, N = N, J = 20000, h2_trait = rep(0.6, 2),
             omega = rep(1, 2), h2_factor = rep(1, 2),
             pi_L = 0.1, pi_theta = 0.1, R_E = R_E)
dat$R
cor(dat$beta_hat[,1]-dat$beta_joint[,1], dat$beta_hat[,2]-dat$beta_joint[,2])
```

`sim_mv` *Simulate multivariate GWAS data*

Description

Simulate multivariate GWAS data

Usage

```
sim_mv(
  N,
  J,
  h2,
  pi,
  G = 0,
  est_s = FALSE,
  R_obs = NULL,
  R_E = NULL,
  R_LD = NULL,
  af = NULL,
  snp_effect_function = "normal",
  snp_info = NULL,
  sporadic_pleiotropy = TRUE,
  pi_exact = FALSE,
  h2_exact = FALSE
)
```

Arguments

N	GWAS sample size. N can be a scalar, vector, or matrix. If N is a scalar, all GWAS have the same sample size and there is no overlap between studies. If N is a vector, each element of N specifies the sample size of the corresponding GWAS and there is no overlap between studies. If N is a matrix, N_ii specifies the sample size of study i and N_ij specifies the number of samples present in both study i and study j. The elements of N must be positive but non-integer values will not generate an error.
J	Number of variants to simulate
h2	A scalar or vector giving the heritability of each trait.
pi	A scalar or vector giving the expected proportion of direct effect SNPs for each trait.
G	Matrix of direct effects. Rows correspond to the 'from' trait and columns correspond to the 'to' trait, so `G[1,2]` is the direct effect of trait 1 on trait 2. G should have 0 on the diagonal.
est_s	If TRUE, return estimates of se(‘beta_hat‘). If FALSE, the exact standard error of ‘beta_hat‘ is returned. Defaults to FALSE.
R_obs	Total observational correlation between traits. R_obs won't impact summary statistics unless there is sample overlap. See Details for default behavior.
R_E	Total correlation of the environmental components only. R_E and R_obs are alternative methods of specifying trait correlation. Use only one of these two options. R_E may be phased out in the future.
R_LD	Optional list of LD blocks. R_LD should have class `list`. Each element of R_LD can be either a) a matrix, b) a sparse matrix (class `dsCMatrix`) or c) an eigen decomposition (class `eigen`). All elements should be correlation matrices, meaning that they have 1 on the diagonal and are positive definite. See Details and vignettes.
af	Optional vector of allele frequencies. If R_LD is not supplied, af can be a scalar, vector or function. If af is a function it should take a single argument (n) and return a vector of n allele frequencies (See Examples). If R_LD is supplied, af must be a vector with length equal to the size of the supplied LD pattern (See Examples).
snp_effect_function	Optional function to generate variant effects. `snp_effect_function` can be a single function or list of functions of length equal to the number of traits (see Details).
snp_info	Optional `data.frame` of variant information to be passed to variant effect functions. If `R_LD` is specified, `snp_info` should have number of rows equal to the size of the supplied LD pattern. Otherwise `snp_info` should have `J` rows.
sporadic_pleiotropy	Allow sporadic pleiotropy between traits. Defaults to TRUE.
pi_exact	If TRUE, the number of direct effect SNPs for each trait will be exactly equal to `round(pi*J)`.
h2_exact	If TRUE, the heritability of each trait will be exactly ‘h2‘.
return_dat	Useful development option, not recommend for general users.

Details

This function generates GWAS summary statistics from a linear SEM specified by the matrix G. The previous "xyz" mode is now deprecated. If you are used to using this function in xyz mode, you can generate the corresponding G using the xyz_to_G function (see examples).

G should be square (#traits by #traits) matrix with 0s on the diagonal. All traits have variance 1, so `G[i,j]^2` is the proportion of variance of trait j explained by the direct effect of trait i. You will get an error if you specify a cyclic DAG. There are also some combinations of DAG and heritability that are impossible and will give an error. For example in a two trait DAG where trait 1 has an effect of x on trait 2, and trait 1 has a heritability of h_1^2, trait 2 must have a heritability of at least $(x^2) * (h_1^2)$.

To generate data with LD, supply `R_LD` which can be a list of matrices, sparse matrices, or eigen-decompositions. These matrices are interpreted as blocks in a block-diagonal LD matrix. The `af` argument must be provided and should be a vector with length equal to the total size of the LD pattern (the sum of the sizes of each block). `R_LD` does not need to have the same size as `J`. The LD pattern will be repeated or subset as necessary to generate the desired number of variants.

If `R_obs` is NULL (default value), we assume that direct environmental effects on each trait are independent and all environmental correlation results from the relationships specified in `G`. Alternatively `R_obs` can be any positive definite correlation matrix.

The `snp_effect_function` argument supplies a custom function to generate standardized direct variant effects. By default, standardized direct effects are generated from a normal distribution. This means that in the default mode, all variants have equal expected heritability explained regardless of allele frequency or other features. The function(s) passed to `snp_effect_function` will override this default. This function should take three arguments, `n`, `sd`, and `snp_info`. The output should be a vector of length `n` with expected total sum of squares equal to `sd^2`. Note: Even though the final effects returned will usually be on the per-allele scale, `snp_effect_function` should return standardized (per-sd) effect sizes.

Value

A list with the following elements:

Simulated effect estimates and standard errors are contained in matrices

+ `beta_hat`: Effect estimates for each trait

+ `se_beta_hat` Standard error of effect estimates, equal to sqrt(1/N_m *Var(G_j)).

+ `s_estimate` Estimate of `se_beta_hat`. Present only if `est_s = TRUE`.

Four matrices contain direct and total marginal and joint SNP-trait associations: + `direct_SNP_effects_marg` and `direct_SNP_effects_joint` give direct effects of SNPs on traits. These are the same if there is no LD. If there is LD, `direct_SNP_effects_marg` is the direct component of the expected marginal association.

+ `beta_marg` and `beta_joint` give the total SNP effects (direct and indirect). These are the same if there is no LD. If there is LD, `beta_marg` is the total expected marginal association. i.e. `beta_marg` is the expected value of `beta_hat`.

Four matrices describe the covariance of traits and the row correlation of effect estimates

+ `Sigma_G` Genetic variance-covariance matrix. Diagonal elements are equal to heritability.

+ `Sigma_E` Environmental variance-covariance matrix.

+ trait_corr Population correlation of traits. trait_corr = Sigma_G + Sigma_E.

+ R Row correlation of beta_hat - beta_marg, equal to trait_corr scaled by the overlap proportion matrix.

Finally,

+ snp_info Is a data frame with variant information. If R_LD was omitted, snp_info contains only the allele frequency of each variant. If R_LD was included, snp_info also contains block and replicate information as well as any information supplied to the snp_info input parameter.

Examples

```
# Two traits with no causal relationship and some environmental correlation
# specify completely overlapping GWAS
N <- matrix(1000, nrow = 2, ncol =2)
R_obs <- matrix(c(1, 0.3, 0.3, 1), nrow = 2, ncol = 2)
dat <- sim_mv(G = 2, N = N, J = 20000, h2 = c(0.4, 0.3), pi = 1000/20000,
              R_obs = R_obs)
dat$R # This is the true correlation of the estimation error of beta_hat
cor(dat$beta_hat - dat$beta_marg) # Should be similar to dat$R

# The af argument can be a scalar, vector, or function.
dat <- sim_mv(N = N, J = 20000, h2 = c(0.4, 0.3), pi = 1000/20000,
              G = 2, R_obs = R_obs, af = function(n){rbeta(n = n, 1, 5)})

# A very simple example with LD
# Use a pattern of two small blocks of LD
A1 <- matrix(0.7, nrow = 10, ncol = 10)
diag(A1) <- 1
A2 <- matrix(0.1, nrow = 6, ncol = 6)
diag(A2) <- 1
# If using LD, af should have the same size as the LD pattern
af <- runif(n = 16)
dat <- sim_mv(N = N, J = 20000, h2 = c(0.4, 0.3), pi = 1000/20000,
              G = 2, R_obs = R_obs, R_LD = list(A1, A2), af = af)

# Use xyz_to_G to generate G from xyz specification
myG <- xyz_to_G(tau_xz = c(0.2, -0.3), tau_yz = c(0.1, 0.25),
        dir_xz = c(1, -1), dir_yz = c(1,1), gamma = 0)
# If N is a scalar or a vector, there is no sample overlap
dat <- sim_mv(N = 10000, J = 20000, h2 = rep(0.4, 4),
              pi = c(500, 500, 1000, 1000)/20000,
              G = myG)
plot(dat$beta_marg[,3], dat$beta_marg[,1])
abline(0, dat$total_trait_effects[3,1])
```

---

sim_mv_determined *Simulate multivariate GWAS Data with Specified Direct Effects*

Description

Simulate multivariate GWAS Data with Specified Direct Effects

Usage

```
sim_mv_determined(
  N,
  direct_SNP_effects_joint,
  geno_scale,
  pheno_sd,
  G = 0,
  est_s = FALSE,
  R_obs = NULL,
  R_E = NULL,
  R_LD = NULL,
  af = NULL
)
```

Arguments

N	GWAS sample size. N can be a scalar, vector, or matrix. If N is a scalar, all GWAS have the same sample size and there is no overlap between studies. If N is a vector, each element of N specifies the sample size of the corresponding GWAS and there is no overlap between studies. If N is a matrix, N_ii specifies the sample size of study i and N_ij specifies the number of samples present in both study i and study j. The elements of N must be positive but non-integer values will not generate an error.
direct_SNP_effects_joint	Matrix of direct variant effects. Should be variants by traits.
geno_scale	Genotype scale of provided effects. Either "allele" or "sd".
pheno_sd	Phenotype standard deviation, a scalar or vector of length number of traits.
G	Matrix of direct effects. Rows correspond to the 'from' trait and columns correspond to the 'to' trait, so `G[1,2]` is the direct effect of trait 1 on trait 2. G should have 0 on the diagonal. Be sure that `G` is on the same scale as the effect sizes.
est_s	If TRUE, return estimates of se(‘beta_hat‘). If FALSE, the exact standard error of ‘beta_hat‘ is returned. Defaults to FALSE.
R_obs	Total observational correlation between traits. R_obs won't impact summary statistics unless there is sample overlap. See Details for default behavior.
R_E	Total correlation of the environmental components only. R_E and R_obs are alternative methods of specifying trait correlation. Use only one of these two options. R_E may be phased out in the future.
R_LD	Optional list of LD blocks. R_LD should have class `list`. Each element of R_LD can be either a) a matrix, b) a sparse matrix (class `dsCMatrix`) or c) an eigen decomposition (class `eigen`). All elements should be correlation matrices, meaning that they have 1 on the diagonal and are positive definite. See Details and vignettes.
af	Optional vector of allele frequencies. If R_LD is not supplied, af can be a scalar, vector or function. If af is a function it should take a single argument (n) and return a vector of n allele frequencies (See Examples). If R_LD is supplied, af must be a vector with length equal to the size of the supplied LD pattern (See Examples).

Details

A wrapper for sim_mv. See ?sim_mv and the "Providing an Exact Set of Direct Effects" section of the Effect Size vignette.

Value

A sim_mv function. See ?sim_mv for details.

Examples

```
G <- matrix(c(0, 0.5, 0, 0), nrow = 2, byrow =TRUE)
my_effects <- matrix(0, nrow = 10, ncol = 2)
my_effects[c(1, 5),1] <- c(-0.008, 0.01)
my_effects[c(3, 6, 9), 2] <- c(-0.02, 0.06, 0.009)
my_effects
# for fun, lets include some sample overlap
N <- matrix(c(40000, 10000, 10000, 20000), nrow = 2)
sim_dat <- sim_mv_determined(N = N,
                             direct_SNP_effects_joint = my_effects,
                             geno_scale = "sd",
                             pheno_sd = 1,
                             G=G,
                             est_s = TRUE)

sim_dat$direct_SNP_effects_joint
sim_dat$beta_joint
sim_dat$Sigma_G
```

---

xyz_to_G *Generate G from XYZ Specification*

Description

Generate G from XYZ Specification

Usage

```
xyz_to_G(tau_xz, tau_yz, dir_xz, dir_yz, gamma)
```

Arguments

dir_xz, dir_yz	Effect direction between Z and X or Y (see details)
gamma	Signed variance of Y explained by X, see details
taux_xz, tau_yz	Effect size between Z and X or Y as signed percent variance explained (see details)

Details

This function generates a matrix G of direct effects corresponding to a model in with variables Y, X, and Z_1, ..., Z_K. There is a causal effect of X on Y given by gamma, which specifies the proportion of the variance of Y explained by X. The K variables Z_1, ..., Z_K can have effects to or from X and Y but do not have direct effects on each other. Vectors `dir_xz` and `dir_yz` specify the direction of these effects with +1 corresponding to "to" effects and -1 corresponding to "from" effects. For example, `dir_xz = c(1, -1)` and `dir_yz = c(1, 1)` would indicate two variables Z_1 and Z_2 with Z_1 being a common cause of both X and Y and Z_2 being a cause of Y and caused by X (Z_2 is a mediator between X and Y). The function will give an error if there is any index with `dir_xz` equal to -1 and `dir_yz` equal 1. This would indicate a variable that is a mediator between Y and X, however, because there is an effect from X to Y assumed, the resulting graph would be cyclic and not allowed.

The inputs 'tau_xz' and 'tau_yz' specify the effect sizes of each of the Z_k variables on or from X and Y. These are given in signed percent variance explained. If we again used `dir_xz = c(1, -1)` and `dir_yz = c(1, 1)` with `tau_xz = c(0.2, -0.3)` and `tau_yz = c(0.1, 0.25)`, this means that the confounder, Z_1 explains 20% of the variance of X and 10% of the variance of Y and both effects are positive. X explains 30% of the variance of the mediate Z_2 with a negative effect direction and Z_2 explains 25% of the variance of Y.

Value

A matrix of direct effects corresponding to variables in the order (Y, X, Z_1, ..., Z_K)

Examples

```
xyz_to_G(tau_xz = c(0.2, -0.3), tau_yz = c(0.1, 0.25),
        dir_xz = c(1, -1), dir_yz = c(1,1), gamma = 0)
# code below will give an error due to specification of a cyclic graph.
## Not run:
xyz_to_G(tau_xz = c(0.2), tau_yz = c(0.1),
        dir_xz = c(1), dir_yz = c(-1), gamma = 0.1)

## End(Not run)
# with gamma = 0, there is no cycle so no error
xyz_to_G(tau_xz = c(0.2), tau_yz = c(0.1),
        dir_xz = c(1), dir_yz = c(-1), gamma = 0)
```

Index