

Package: simulateGP (via r-universe)

October 8, 2026

Title Functions for Simulating Genotype-Phenotype Relationships

Version 0.1.3

Description Functions for simulating SNP effects on phenotypes.
Functions for simulating disease risk on probability and liability scales. Simulations to quickly obtain estimates of effect sizes for simulated genotype-phenotypes.

License MIT + file LICENSE

URL <https://explodecomputer.github.io/simulateGP/>,
<https://github.com/explodecomputer/simulateGP>,
<https://mrcieu.r-universe.dev/simulateGP>

BugReports <https://github.com/explodecomputer/simulateGP/issues>

Depends R (>= 4.0.0)

Imports data.table, dplyr, genetics.binaRies, ggplot2, graphics,
gwasglue2, ieugwasr, magrittr, MASS, parallel, pbapply, rlang,
stats, TwoSampleMR

Suggests covr, dagitty, ggdag, jsonlite, knitr, mvtnorm, rmarkdown,
systemfit, testthat, tidyverse

VignetteBuilder knitr

Remotes mrcieu/genetics.binaRies, mrcieu/gwasglue2, mrcieu/ieugwasr,
mrcieu/TwoSampleMR

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.1.0

Encoding UTF-8

LazyData true

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev

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

Date/Publication 2026-10-01 08:38:24 UTC

RemoteUrl <https://github.com/explodecomputer/simulateGP>

RemoteRef refs/pull/6/head

RemoteSha dd16b78fcb43dabd3d8a83baaec39f4f4faaec9f

Contents

simulateGP-package	3
add_u	4
allele_frequency	4
arbitrary_map	5
ascertain_samples	5
choose_effects	6
contingency	6
correlated_binomial	7
create_system	7
estimate_system_effects	9
expected_mse	9
expected_se	10
expected_ssx	10
fast_assoc	11
generate_gwas_params	11
generate_gwas_ss	12
generate_gwas_ss_1	13
generate_ldobj	13
get_effs	14
get_ld	15
get_population_allele_frequency	15
get_regions_from_ldobjdir	16
gwas	16
gx_to_gp	17
hap_freqs	17
init_parameters	18
ldetect	19
lor_to_rsqr	19
make_genotype	20
make_mvdat	20
make_phen	21
merge_exp_out	21
range01	22
read_ldobjdir	22
recode_dat	23
recode_dat_intercept	23
recode_dat_simple	24
risk_cross_plot	24
risk_simulation	25
sample_beta	25
sample_system_effects	26
simulate_genotype	26
simulate_haplotypes	27
simulate_population	27
stuff	28
summary_set	28

<i>simulateGP-package</i>	3
test_ldobj	29
test_system	29
y_to_binary	30
Index	31

simulateGP-package	<i>simulateGP: Functions for Simulating Genotype-Phenotype Relationships</i>
--------------------	--

Description

This package is a collection of utilities relating to simulating genotype phenotype maps. The simulations are largely used for Mendelian randomisation, and link up with the TwoSampleMR R package.

Details

Full documentation available here: <https://explodecomputer.github.io/simulateGP>

Author(s)

Maintainer: Gibran Hemani <explodecomputer@gmail.com> ([ORCID](#))

Authors:

- Gibran Hemani <explodecomputer@gmail.com> ([ORCID](#))
- John Ferguson <john.ferguson@nuigalway.ie>
- Rita Rasteiro <rita.rasteiro@bristol.ac.uk> ([ORCID](#))

See Also

Useful links:

- <https://explodecomputer.github.io/simulateGP/>
- Report bugs at <https://github.com/explodecomputer/simulateGP/issues>

add_u	<i>Add confounder variables and their instruments</i>
-------	---

Description

Add confounder variables and their instruments

Usage

```
add_u(parameters, nsnp_u, var_u.x, var_u.y, var_gu.u)
```

Arguments

parameters	Output from init_parameters
nsnp_u	nsnp_u
var_u.x	var_u.x
var_u.y	var_u.y
var_gu.u	var_gu.u

Value

List of model parameters

allele_frequency	<i>Estimate allele frequency from SNP</i>
------------------	---

Description

Estimate allele frequency from SNP

Usage

```
allele_frequency(g)
```

Arguments

g	Vector of 0/1/2
---	-----------------

Value

Allele frequency

arbitrary_map	Create an arbitrary map
---------------	-------------------------

Description

Create an arbitrary map

Usage

```
arbitrary_map(af)
```

Arguments

af	vector of allele frequencies
----	------------------------------

Value

data frame with snp name, chr, pos, ea, oa

ascertain_samples	Ascertain some proportion of cases and controls from binary phenotype
-------------------	---

Description

Ascertain some proportion of cases and controls from binary phenotype

Usage

```
ascertain_samples(d, prop_cases)
```

Arguments

d	Vector of 1/0
prop_cases	Proportion of 1s to retain

Value

Array of IDs

choose_effects	<i>Get vector of effects that explain some amount of variance</i>
----------------	---

Description

Get vector of effects that explain some amount of variance

Usage

```
choose_effects(nsnp, totvar, sqrt = TRUE, mua = 0)
```

Arguments

nsnp	Number of SNPs
totvar	Total variance explained by all SNPs
sqrt	Output effect sizes in terms of standard deviations. Default=TRUE
mua	Constant term to be added to effects. Default = 0

Value

Vector of effects

contingency	<i>Obtain 2x2 contingency table from marginal parameters and odds ratio</i>
-------------	---

Description

Columns are the case and control frequencies Rows are the frequencies for allele 1 and allele 2

Usage

```
contingency(af, prop, odds_ratio, eps = 1e-15)
```

Arguments

af	Allele frequency of effect allele
prop	Proportion of cases
odds_ratio	Odds ratio
eps	tolerance. Default = 1e-15

Value

2x2 contingency table as matrix

correlated_binomial	<i>Simulate two correlated binomial variables</i>
---------------------	---

Description

Simulate two correlated binomial variables

Usage

```
correlated_binomial(nid, p1, p2, rho, n = 2, round = TRUE, print = FALSE)
```

Arguments

nid	Number of samples
p1	Frequency 1
p2	Frequency 2
rho	Target correlation
n	Binomial parameter, should be 2 (default) for genotypes
round	Round or not Default=TRUE
print	Print or not Default=FALSE

Value

Matrix

create_system	<i>Wrapper for simulation pipeline</i>
---------------	--

Description

Based on the parameters specified this function will call `init_parameters`, `add_u`, `sample_system_effects`, `simulate_population` and `estimate_system_effects`. A separate population is generated for each phenotype (x, y and each of u) to allow 2SMR and PRS analyses

Usage

```
create_system(  
  nidx,  
  nidy,  
  nidu = 0,  
  nu = 0,  
  na = 0,  
  nb = 0,  
  var_x.y,
```

```

    nsnp_x,
    var_gx.x,
    var_gx.y = 0,
    mu_gx.y = 0,
    prop_gx.y = 1,
    nsnp_y = 0,
    var_gy.y = 0,
    var_gy.x = 0,
    mu_gy.x = 0,
    prop_gy.x = 1
  )

```

Arguments

nidx	number of individuals for x
nidy	number of individuals for y
nidu	Default=0 If 0 then don't simulate separate populations for the u variables
nu	Default=0 Number of confounders influencing x and y
na	Default=0 Number of traits upstream of x
nb	Default=0 Number of traits upstream of y
var_x.y	var in y explained by x
nsnp_x	nsnp influencing x
var_gx.x	variance in x explained by instruments for x
var_gx.y	Default=0
mu_gx.y	Default=0
prop_gx.y	Default=1
nsnp_y	Default=0
var_gy.y	Default=0
var_gy.x	Default=0
mu_gy.x	Default=0
prop_gy.x	Default=1

Value

List of effects across system

`estimate_system_effects`*Estimate the effects of all SNPs on all phenotypes*

Description

Estimate the effects of all SNPs on all phenotypes

Usage

```
estimate_system_effects(sim)
```

Arguments

`sim` Output from `simulate_population`

Value

Lists of SNP-trait effect estimates

`expected_mse`*Calculate expected MSE*

Description

Calculate expected MSE

Usage

```
expected_mse(beta, af, vy)
```

Arguments

`beta` array of effect sizes
`af` array of allele frequencies
`vy` variance of y

Value

Numeric

expected_se	<i>Expected se given beta, af, n and vy</i>
-------------	---

Description

$$se = \sqrt{\sigma_e^2 / ss(x)}$$

Usage

```
expected_se(beta, af, n, vy)
```

Arguments

- beta array of effect sizes
- af array of allele frequencies
- n sample size
- vy variance of y

Value

array of standard errors

expected_ssx	<i>Calculate expected SSX</i>
--------------	-------------------------------

Description

Calculate expected SSX

Usage

```
expected_ssx(af, n)
```

Arguments

- af array of allele frequencies
- n sample size

Value

Numeric

fast_assoc	<i>Get summary statistics in simple linear regression</i>
------------	---

Description

Get summary statistics in simple linear regression

Usage

```
fast_assoc(y, x)
```

Arguments

y	Vector of dependent variable
x	Vector of independent variable

Value

List

generate_gwas_params	<i>Generate SNP effects given MAF, h2 and selection</i>
----------------------	---

Description

Generate SNP effects given MAF, h2 and selection

Usage

```
generate_gwas_params(map, h2, S = 0, Pi = 1)
```

Arguments

map	Data frame containing at least af allele frequency and snp SNP columns. SNPs must be unique, af must be between 0 and 1. Optionally also include the chr, pos, ref, alt columns if using LD-aware simulations
h2	Variance explained by all SNPs
S	Selection coefficient on trait. Default = 0
Pi	Proportion of variants that have an effect - sampled randomly. Default=1

Value

data frame

generate_gwas_ss

Modify SNP effects to account for LD

Description

After generating the set of causal effects at each SNP, use an LD correlation matrix to transform the effects to reflect the correlation structure at the SNPs. Note if running many repeats, only need to generate the LD-modified params once and then can repeatedly re-sample using generate_gwas_ss

Usage

```
generate_gwas_ss(
  params,
  nid,
  vy = 1,
  minmaf = 0.001,
  ld = NULL,
  ldobj = NULL,
  ldobjlist = NULL,
  ldobjfiles = NULL,
  ldobjdir = NULL,
  gwasglue2 = FALSE,
  nthreads = 1
)
```

Arguments

params	Output from generate_gwas_params
nid	sample size
vy	Variance of trait
minmaf	minimum allowed maf. default=0.01 to prevent instability
ld	LD correlation matrix. Must be same dimension as params
ldobj	LD objects (e.g. see test_ldobj)
ldobjlist	List of LD objects
ldobjfiles	Array of filenames containing LD object files (e.g. see generate_ldobj)
ldobjdir	Directory containing output from generate_ldobj
gwasglue2	Logical, creates a gwasglue2 SummarySet when @param ldobj is not NULL (default FALSE).
nthreads	Number of threads (can be slow for complete GWAS and large LD regions)

Value

Updated params

generate_gwas_ss_1	<i>Create a GWAS summary dataset</i>
--------------------	--------------------------------------

Description

Determines SE and generates effect estimates given input parameters

Usage

```
generate_gwas_ss_1(params, nid, vy = 1, minmaf = 0.01, r = NULL)
```

Arguments

params	Output from generate_gwas_params
nid	sample size
vy	Variance of trait
minmaf	minimum allowed maf. default=0.01 to prevent instability
r	LD correlation matrix. If NULL (default) then creates LD unaware sampling errors

Value

list of data frames

generate_ldobj	<i>Generate LD matrix objects from reference panel</i>
----------------	--

Description

Creates a set of ldobj files, each corresponding to a single independent LD region from the reference panel. It also generates a map file.

Usage

```
generate_ldobj(
  outdir,
  bfile,
  regions,
  plink_bin = genetics.binaRies::get_plink_binary(),
  nthreads = 1
)
```

Arguments

outdir	Directory in which to store the lobj files
bfile	Binary plink dataset
regions	A data frame containing the independent regions (see data(ldetect))
plink_bin	Plink executable. Default=genetics.binaRies::get_plink_binary()
nthreads	How many threads. Default=1

Value

map file

get_effs	<i>Get effs for two traits and make dat format</i>
----------	--

Description

Get effs for two traits and make dat format

Usage

```
get_effs(x, y, g, xname = "X", yname = "Y")
```

Arguments

x	Vector of exposure phenotype
y	Vector of outcome phenotype
g	Matrix of genotypes
xname	xname
yname	yname

Value

Data frame

get_ld	<i>Get LD matrix for a specified region from bfile reference panel</i>
--------	--

Description

Get LD matrix for a specified region from bfile reference panel

Usage

```
get_ld(
  chr,
  from,
  to,
  bfile,
  plink_bin = genetics.binaRies::get_plink_binary(),
  nref = NULL
)
```

Arguments

chr	Chromosome
from	from bp
to	to bp
bfile	LD reference panel
plink_bin	Plink binary default=genetics.binaRies::get_plink_binary()
nref	Sample size of reference panel

Value

List of LD matrix and map info including MAF

get_population_allele_frequency	<i>Estimate the allele frequency in population from case/control summary data</i>
---------------------------------	---

Description

Estimate the allele frequency in population from case/control summary data

Usage

```
get_population_allele_frequency(af, prop, odds_ratio, prevalence)
```

Arguments

af	Effect allele frequency (or MAF)
prop	Proportion of samples that are cases
odds_ratio	Odds ratio
prevalence	Population disease prevalence

Value

Population allele frequency

`get_regions_from_ldobjdir`

Determine regions from LD file

Description

Determine regions from LD file

Usage

```
get_regions_from_ldobjdir(ldobjdir)
```

Arguments

ldobjdir	Directory containing output from <code>generate_ldobj</code>
----------	--

Value

Data frame

`gwas`

Perform association of many SNPs against phenotype

Description

Perform association of many SNPs against phenotype

Usage

```
gwas(y, g, logistic = FALSE)
```

Arguments

y	Vector of phenotypes
g	Matrix of genotypes
logistic	Use logistic regression (much slower)? Default=FALSE

Value

Data frame

gx_to_gp	<i>Translate risk from liability to probability scale</i>
----------	---

Description

Translate risk from liability to probability scale

Usage

```
gx_to_gp(gx, h2x, prev)
```

Arguments

gx	score on liability scale
h2x	Disease heritability on liability scale
prev	Prevalence

Value

Vector of disease probabilities

hap_freqs	<i>Estimate haplotype frequencies for two loci</i>
-----------	--

Description

Estimate haplotype frequencies for two loci

Usage

```
hap_freqs(r, p1, p2)
```

Arguments

r	Required LD r
p1	Freq 1
p2	Freq 2

Value

vector

init_parameters	<i>Choose initial parameters for direct effects on X and Y</i>
-----------------	--

Description

Choose initial parameters for direct effects on X and Y

Usage

```
init_parameters(  
  nsnp_x,  
  var_gx.x,  
  var_x.y,  
  var_gx.y = 0,  
  nsnp_y = 0,  
  var_gy.y = 0,  
  mu_gx.y = 0,  
  var_gy.x = 0,  
  mu_gy.x = 0,  
  prop_gy.x = 1,  
  prop_gx.y = 1  
)
```

Arguments

nsnp_x	nsnp_x
var_gx.x	var_gx.x
var_x.y	var_x.y
var_gx.y	var_gx.y=0
nsnp_y	nsnp_y=0
var_gy.y	var_gy.y=0
mu_gx.y	mu_gx.y=0
var_gy.x	var_gy.x=0
mu_gy.x	mu_gy.x=0
prop_gy.x	prop_gy.x=1
prop_gx.y	prop_gx.y=1

Value

List of model parameters

ldetect	<i>Data frame of independent LD regions</i>
---------	---

Description

Taken from Berisa and Pickrell (2016) - <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4731402/>
 - a list of independent LD regions for Africans, Europeans and Asians. All in build hg19

Usage

```
ldetect
```

Format

A data frame with 5731 rows and 4 columns

chr chromosome

start start position

stop stop position

pop Population

Source

<https://bitbucket.org/nygcresearch/ldetect-data>](<https://bitbucket.org/nygcresearch/ldetect-data/src/master/>)

lor_to_rsq	<i>Estimate proportion of variance of liability explained by SNP in general population</i>
------------	--

Description

This uses equation 10 in Genetic Epidemiology 36 : 214–224 (2012)

Usage

```
lor_to_rsq(b, af, ncase, ncontrol, prevalence, model = "logit")
```

Arguments

b	Log odds ratio
af	allele frequency
ncase	Number of cases
ncontrol	number of controls
prevalence	prevalence
model	Is the effect size estimated in "logit" (default) or "probit" model

Value

Rsq

make_geno

*Create genotype matrix***Description**

Create genotype matrix

Usage

make_geno(nid, nsnp, af)

Arguments

nid	Number of samples
nsnp	Number of SNPs
af	Allele frequency of all SNPs (all SNPs have the same allele frequency)

Value

Matrix of genotypes, rows = individuals, columns = snps

make_mvdat

*Take several exposures and one outcome and make the data required for multivariable MR***Description**

Take several exposures and one outcome and make the data required for multivariable MR

Usage

make_mvdat(exposures, y, g)

Arguments

exposures	List of exposure vectors
y	Vector of outcomes
g	Matrix of genotypes

Value

mv_harmonise_data output

make_phen

*Simulate variable based on the influences of other variables***Description**

I need a function that takes some variables, and an effect size for each of them, and this creates a new variable where $y = Xb + e$, where X is the matrix of independent variables, b is their effect sizes, and e is a noise term. By default it's easier if b relates to the variance explained by each vector in X , and e has variance of $1 - \text{sum}(b)$.

Usage

```
make_phen(effs, indep, vy = 1, vx = rep(1, length(effs)), my = 0)
```

Arguments

effs	An array of effect sizes that the variables in indep have on the variable that you are simulating
indep	A matrix of variables, rows = Samples and columns = variables that have an influence on the variable that you are simulating
vy	What variance the output should have. Default = 1, meaning effs relate to the signed rsq of the influence of the indep variables on the outcome
vx	What variance the indep variables should have. Default is to set to 1, meaning that the effects are the signed variance explained
my	mean value of y to be output

Value

Vector of y values

merge_exp_out

*Organise outputs from gwas into harmonised dat format***Description**

Organise outputs from gwas into harmonised dat format

Usage

```
merge_exp_out(gwasx, gwasy, xname = "X", yname = "Y")
```

Arguments

gwasx	Output from gwas
gwasy	Output from gwas
xname	exposure name
yname	outcome name

Value

data frame

range01	<i>Scale variable to have range between 0 and 1</i>
---------	---

Description

Scale variable to have range between 0 and 1

Usage

range01(x)

Arguments

x	Vector
---	--------

Value

Vector

read_ldobjdir	<i>Read in LD objects into list</i>
---------------	-------------------------------------

Description

Read in LD objects into list

Usage

read_ldobjdir(ldobjdir, nthreads = 1)

Arguments

ldobjdir	Directory containing output from generate_ldobj
nthreads	Number of threads. Default=1

Value

List of ldobj

recode_dat	<i>Recode data to make every effect on x positive</i>
------------	---

Description

Can use simple method or by pivoting around intercept

Usage

```
recode_dat(dat, method = "intercept")
```

Arguments

dat	Output from get_effs
method	Default 'intercept'. Alternatively can specify 'simple'

Value

Data frame

recode_dat_intercept	<i>Intercept recoding to have every effect on x positive</i>
----------------------	--

Description

Tries to avoid issue of recoding by finding intercept and pivoting negative g-x associations around intercept

Usage

```
recode_dat_intercept(dat)
```

Arguments

dat	Output from get_effs
-----	----------------------

Value

Data frame

recode_dat_simple	<i>Simple recoding to have every effect on x positive</i>
-------------------	---

Description

Simple recoding to have every effect on x positive

Usage

```
recode_dat_simple(dat)
```

Arguments

dat	Output from get_effs
-----	----------------------

Value

Data frame

risk_cross_plot	<i>Plot liability vs probability disease risk</i>
-----------------	---

Description

Plot liability vs probability disease risk

Usage

```
risk_cross_plot(
  x,
  o,
  xlab = "Values (low to high)",
  ylab = "",
  title = "",
  xname = "GRS",
  oname = "Disease"
)
```

Arguments

x	Disease risk on liability scale
o	Disease risk on probability scale
xlab	default="Values (low to high)"
ylab	default=""
title	default=""
xname	Name of liability. Default="GRS"
oname	Name of disease. Default="Disease"

Value

ggplot

risk_simulation	<i>Make simulation to compare disease and liability scales</i>
-----------------	--

Description

Compares the liability and probability of disease under two scenarios - where all SNPs are known, and where only some SNPs are known

Usage

```
risk_simulation(G, eff, prevalence, prop_discovered)
```

Arguments

G	Matrix of genotypes
eff	SNP effects on liability scale
prevalence	Disease prevalence
prop_discovered	Proportion of SNPs discovered

Value

Data frame

sample_beta	<i>Sample beta values given standard errors</i>
-------------	---

Description

Sample beta values given standard errors

Usage

```
sample_beta(beta, se, r = NULL, af = NULL)
```

Arguments

beta	array of beta values - i.e. the true coefficients. If using the correlation matrix r then this should be an LD-aware set of expected beta values.
se	array of se values
r	matrix of LD correlations amongst the SNPs. If NULL (default) then assumes no LD
af	array of allele frequencies. Must be non-null if r is non-null.

Value

array of beta hats

sample_system_effects	<i>Sample the actual effects based on initial parameters</i>
-----------------------	--

Description

Sample the actual effects based on initial parameters

Usage

sample_system_effects(parameters)

Arguments

parameters Output from init_parameters or add_u

Value

List of effect parameters

simulate_geno	<i>Simulate genotypes from haplotypes</i>
---------------	---

Description

Simulate genotypes from haplotypes

Usage

simulate_geno(nid, r, p1, p2)

Arguments

nid	Number of samples
r	Desired LD r
p1	Freq 1
p2	Freq 2

Value

Matrix

simulate_haplotypes	<i>Simulate haplotypes of two loci</i>
---------------------	--

Description

Simulate haplotypes of two loci

Usage

```
simulate_haplotypes(nid, r, p1, p2)
```

Arguments

nid	Number of samples
r	Desired LD r
p1	Freq 1
p2	Freq 2

Value

Matrix

simulate_population	<i>Simulate individual level data from initial parameters</i>
---------------------	---

Description

Simulate individual level data from initial parameters

Usage

```
simulate_population(parameters, nid)
```

Arguments

parameters	Output from init_parameters or add_u
nid	Sample size to generate

Value

List of matrices and vectors that represent individual level data

stuff	General funcs
Description	
General funcs	
summary_set	Wrapper for generating a summary set

Description

Allows arbitrary sample overlap

Usage

```
summary_set(  
  beta_gx,  
  beta_gy,  
  af,  
  n_gx,  
  n_gy,  
  n_overlap,  
  cor_xy,  
  prev_y = NA,  
  sigma_x = 1,  
  sigma_y = 1  
)
```

Arguments

beta_gx	Array of true effects on x
beta_gy	Array of true effects on y
af	Array of effect allele frequencies
n_gx	Sample size of g-x association
n_gy	Sample size of g-y association
n_overlap	Number of overlapping samples
cor_xy	Observational correlation between x and y
prev_y	Disease prevalence of y. Default = NA (in which case treated as continuous)
sigma_x	SD of x. Default=1
sigma_y	SD of y. Default=1

Value

Data frame of summary statistics for x and y

test_ldobj	<i>Create test LD object</i>
------------	------------------------------

Description

Create test LD object

Usage

```
test_ldobj(nsnp, chunksize)
```

Arguments

nsnp	Number of SNPs
chunksize	Chunksize for splitting

Value

list of chunks, which each contain map and LD matrix

test_system	<i>Apply MR tests to system</i>
-------------	---------------------------------

Description

Apply MR tests to system

Usage

```
test_system(ss, id = "test")
```

Arguments

ss	Output from create_syste,
id	string denoting simulation ID

Value

List

y_to_binary	<i>Convert continuous trait to binary</i>
-------------	---

Description

Convert continuous trait to binary

Usage

```
y_to_binary(y, prevalence = NULL, threshold = NULL)
```

Arguments

y	Phenotype vector
prevalence	Disease prevalence. Default = NULL
threshold	Disease threshold Default = NULL

Value

Vector of binary trait

Index

* datasets

ldetect, [19](#)

add_u, [4](#)
allele_frequency, [4](#)
arbitrary_map, [5](#)
ascertain_samples, [5](#)

choose_effects, [6](#)
contingency, [6](#)
correlated_binomial, [7](#)
create_system, [7](#)

estimate_system_effects, [9](#)
expected_mse, [9](#)
expected_se, [10](#)
expected_ssx, [10](#)

fast_assoc, [11](#)

generate_gwas_params, [11](#)
generate_gwas_ss, [12](#)
generate_gwas_ss_1, [13](#)
generate_ldobj, [13](#)
get_effs, [14](#)
get_ld, [15](#)
get_population_allele_frequency, [15](#)
get_regions_from_ldobjdir, [16](#)
gwas, [16](#)
gx_to_gp, [17](#)

hap_freqs, [17](#)

init_parameters, [18](#)

ldetect, [19](#)
lor_to_rsqr, [19](#)

make_genotype, [20](#)
make_mvdat, [20](#)
make_phen, [21](#)

merge_exp_out, [21](#)

range01, [22](#)
read_ldobjdir, [22](#)
recode_dat, [23](#)
recode_dat_intercept, [23](#)
recode_dat_simple, [24](#)
risk_cross_plot, [24](#)
risk_simulation, [25](#)

sample_beta, [25](#)
sample_system_effects, [26](#)
simulate_genotype, [26](#)
simulate_haplotypes, [27](#)
simulate_population, [27](#)
simulateGP (simulateGP-package), [3](#)
simulategp (simulateGP-package), [3](#)
simulateGP-package, [3](#)
stuff, [28](#)
summary_set, [28](#)

test_ldobj, [29](#)
test_system, [29](#)

y_to_binary, [30](#)