Package ‘FieldSimR’

January 31, 2023

Title Simulation of Plot-Level Data in Plant Breeding Field Trials

Version 1.0.0

Date 2022-09-19

Maintainer Christian Werner `<werner.christian@proton.me>`

Description Simulates plot data in plant breeding field trials for multiple traits in multiple environments. Its core function simulates spatially correlated plot errors across correlated traits using bivariate interpolation or a two-
dimensional autoregressive process of order one (AR1:AR1).
'FieldSimR' then combines this spatial error with random measurement error at a user-defined ratio. The simulated plot errors can be combined with genetic values (e.g. true, simulated or predicted) to generate plot-
level phenotypes. 'FieldSimR' provides wrapper functions to simulate the
genetic values for multiple traits in multiple environments using the 'R' package 'AlphaSimR'.

License GPL (>= 3)

URL https://github.com/crWerner/fieldsimr

Encoding UTF-8

LazyData true

Imports fields, interp, matrixcalc, mbend

Suggests AlphaSimR

RoxygenNote 7.2.3

Depends R (>= 2.10)

R topics documented:

`compsym_asr_input`	*Genetic values based on a compound symmetry model for GxE interaction using 'AlphaSimR' - Input parameters*

Description

Creates a list of input simulation parameters for 'AlphaSimR' to simulate genetic values for multiple traits in multiple environments based on a compound symmetry model for genotype-by-environment (GxE) interaction.

By default, 'AlphaSimR' does not support complex models for GxE interaction. However, its functionality to simulate correlated genetic values can be utilised for this purpose by providing the required variance structures. `compsym_asr_input` is a wrapper function to construct the variance structures required to simulate GxE interaction in 'AlphaSimR' based on a compound symmetry model. This function assumes a separable structure between traits and environments. It is also used in combination with the wrapper function compsym_asr_output.

Usage

```
compsym_asr_input(
  n_envs,
  n_traits,
  mean,
  var,
  rel_main_eff_A,
  cor_A = NULL,
  mean_DD = NULL,
  var_DD = NULL,
  rel_main_eff_DD = NULL,
  cor_DD = NULL,
  rel_AA = NULL,
  rel_main_eff_AA = NULL,
  cor_AA = NULL
)
```

Arguments

`n_envs`	Number of environments to be simulated. A minimum of two environments is required.
`n_traits`	Number of traits to be simulated.
`mean`	A vector of mean genetic values for each trait-by-environment combination (ordered as environments within traits). Simulated traits can have a different mean for each environment. If the length of `mean` corresponds to `n_traits`, all traits will be assigned the same mean for each environment.
`var`	A vector of genetic variances for each trait. Simulated traits are restricted by the compound symmetry model to having the same variance for each environment (i.e., main effect variance + GxE interaction variance) and the same covariance between each pair of environments (main effect variance). **Note:** when `useVarA = TRUE` is specified in 'AlphaSimR' (default) the values in `var` represent the `additive` genetic variances, otherwise they will represent the `total` (additive + non-additive) genetic variances.
`rel_main_eff_A`	A vector defining the magnitude of the additive main effect variance relative to the additive main effect + GxE interaction variance for each trait. If only one value is provided and `n_traits` > 1, all traits will be assigned the same value. **Note:** `0` < `rel_main_eff_A` < 1.
`cor_A`	A matrix of additive genetic correlations between more than one trait. If not defined and `n_traits` > 1, a diagonal matrix is constructed.

`mean_DD` A vector of mean dominance degrees for each trait-by-environment combination (ordered as environments within traits), similar to `mean`. By default, `mean_DD = NULL` and dominance is not simulated.

`var_DD` A vector of dominance degree variances for each trait. Simulated traits have the same dominance degree variance for each environment and the same dominance degree covariance between each pair of environments (similar to `var`). By default, `var_DD = NULL`.

`rel_main_eff_DD` A vector defining the magnitude of the dominance degree main effect variance relative to the main effect + GxE interaction variance for each trait (similar to `rel_main_eff_A`)
Note: `0 < rel_main_eff_DD < 1`. By default, `rel_main_eff_DD = NULL`.

`cor_DD` A matrix of dominance degree correlations between more than one trait (similar to `cor_A`). If not defined and `n_traits > 1`, a diagonal matrix is constructed. By default, `cor_DD = NULL`.

`rel_AA` A vector defining the magnitude of additive-by-additive (epistatic) variance relative to the additive genetic variance for each trait, that is in a diploid organism with allele frequency 0.5. Simulated traits have the same epistatic variance for each environment and the same epistatic covariance between each pair of environments (similar to `var`). By default, `rel_AA = NULL` and epistasis is not simulated.

`rel_main_eff_AA` A vector defining the magnitude of the epistatic main effect variance relative to the main effect + GxE interaction variance for each trait (similar to `rel_main_eff_A`).
Note: `0 < rel_main_eff_AA < 1`. By default, `rel_main_eff_AA = NULL`.

`cor_AA` A matrix of epistatic correlations between more than one trait (similar to `cor_A`). If not defined and `n_traits > 1`, a diagonal matrix is constructed. By default, `cor_AA = NULL`.

Details

Note: 'AlphaSimR' can simulate different biological effects (see: `SimParam`).

- For additive traits use `addTraitA()`.
- For additive + dominance traits use `addTraitAD()`.
- For additive + epistatic traits use `addTraitAE()`.
- For additive + dominance + epistatic traits use `addTraitADE()`.

If non-additive effects are to be simulated, check the `useVarA` argument of these functions.

Value

A list containing input parameters for 'AlphaSimR', which is used to simulate correlated genetic effects based on a compound symmetry model.

Examples

```
# Simulation of genetic values in 'AlphaSimR' for two additive + dominance traits tested in
# three environments based on a compound symmetry model for GxE interaction.

# 1. Define the genetic architecture of the simulated traits.
# Mean genetic values and mean dominance degrees for trait 1 in all 3 environments and trait 2
```

```
# in all 3 environments.
mean <- c(1, 3, 2, 80, 70, 100) # Trait 1 x 3 environments, trait 2 x 3 environments.
mean_DD <- c(0.1, 0.4) # Trait 1 and 2, same values set in 3 environments for each trait.

# Additive genetic variances (useVarA = TRUE) and dominance degree variances for traits 1 and 2.
var <- c(0.2, 10)
var_DD <- c(0.1, 0.2)

# Relative magnitude of the additive and dominance degree main effect variance for traits 1 and 2.
rel_main_eff_A <- c(0.4, 0.6) # Different values set for traits 1 and 2.
rel_main_eff_DD <- 0.8 # Same value set for traits 1 and 2.

# Additive and dominance degree correlations between traits 1 and 2.
cor_A <- matrix(c(1.0, 0.3, 0.3, 1.0), ncol = 2) # Additive correlation matrix.
cor_DD <- diag(2) # Assuming independence between traits.

input_asr <- compsym_asr_input(
  n_envs = 3,
  n_traits = 2,
  mean = mean,
  var = var,
  rel_main_eff_A = rel_main_eff_A,
  cor_A = cor_A,
  mean_DD = mean_DD,
  var_DD = var_DD,
  rel_main_eff_DD = rel_main_eff_DD,
  cor_DD = cor_DD
)
```

`compsym_asr_output` *Genetic values based on a compound symmetry model for GxE interaction using 'AlphaSimR' - Simulation of genetic values*

Description

Creates a data frame of correlated genetic values for multiple traits in multiple environments based on a compound symmetry model for genotype-by-environment (GxE) interaction. This function requires an 'AlphaSimR' population object generated using the compsym_asr_input function.

Usage

```
compsym_asr_output(pop, n_envs, n_reps, n_traits, effects = FALSE)
```

Arguments

`pop`	An 'AlphaSimR' population object (`Pop-class` or `HybridPop-class`) generated using compsym_asr_input.
`n_envs`	Number of simulated environments (same as used in compsym_asr_input).
`n_reps`	A vector defining the number of complete replicates in each environment. If only one value is provided and `n_traits` > 1, all environments will be assigned the same number of replicates.
`n_traits`	Number of simulated traits (same as used in compsym_asr_input).

`effects` When TRUE, a list is returned with additional entries containing the total (additive + dominance + epistatic) main effects and GxE interaction effects for each trait-by-environment combination. By default, effects = FALSE.

Value

A data-frame containing the environment id, replicate number, genotype id, and simulated genetic values for each trait. When `effects = TRUE`, a list is returned with additional entries containing the total (additive + dominance + epistatic) main effects and GxE interaction effects for each trait-by-environment combination.

Examples

```
# Simulation of genetic values in 'AlphaSimR' for two additive + dominance traits tested in
# three environments based on a compound symmetry model for GxE interaction.

# 1. Define the genetic architecture of the simulated traits.
# Mean genetic values and mean dominance degrees for trait 1 in all 3 environments and trait 2
# in all 3 environments.
mean <- c(1, 3, 2, 80, 70, 100) # Trait 1 x 3 environments, trait 2 x 3 environments.
mean_DD <- c(0.1, 0.4) # Trait 1 and 2, same values set in 3 environments for each trait.

# Additive genetic variances (useVarA = TRUE) and dominance degree variances for traits 1 and 2.
var <- c(0.2, 10)
var_DD <- c(0.1, 0.2)

# Relative magnitude of additive and dominance degree main effect variance for traits 1 and 2.
rel_main_eff_A <- c(0.4, 0.6) # Different values set for traits 1 and 2.
rel_main_eff_DD <- 0.8 # Same value set for traits 1 and 2.

# Additive and dominance degree correlations between traits 1 and 2.
cor_A <- matrix(c(1.0, 0.3, 0.3, 1.0), ncol = 2) # Additive correlation matrix.
cor_DD <- diag(2) # Assuming independence between traits.

input_asr <- compsym_asr_input(
  n_envs = 3,
  n_traits = 2,
  mean = mean,
  var = var,
  rel_main_eff_A = rel_main_eff_A,
  cor_A = cor_A,
  mean_DD = mean_DD,
  var_DD = var_DD,
  rel_main_eff_DD = rel_main_eff_DD,
  cor_DD = cor_DD
)


# 2. Use input_asr to simulate genetic values in 'AlphaSimR' based on a compound symmetry model
# for GxE interaction.

library("AlphaSimR")
FOUNDERPOP <- quickHaplo(
  nInd = 100,
  nChr = 6,
  segSites = 100
)
```

```
SP <- SimParam$new(FOUNDERPOP)

SP$addTraitAD(
  nQtlPerChr = 100,
  mean = input_asr$mean,
  var = input_asr$var,
  meanDD = input_asr$mean_DD,
  varDD = input_asr$var_DD,
  corA = input_asr$cor_A,
  corDD = input_asr$cor_DD,
  useVarA = TRUE
)

# By default, the value provided in 'var' represents the additive variance.
# If useVarA=FALSE, 'var' represents the total genetic variance.

pop <- newPop(FOUNDERPOP)


# 3. Create a data frame containing the simulated genetic values for the two traits in the
# three environments.

n_reps <- c(2, 3, 2) # Vector containing the number of complete replicates in each environment.

gv_df <- compsym_asr_output(
  pop = pop,
  n_envs = 3,
  n_reps = n_reps,
  n_traits = 2,
  effects = TRUE
)
```

`df_error_bivar` *Field trial error example data frame*

Description

An example data frame of simulated field trial plot errors for two traits tested in three environments. The data frame was generated using the function field_trial_error with bivariate interpolation.

Usage

```
df_error_bivar
```

Format

A data frame with 700 rows and 5 columns:

env Environment id

block Block id

col Column id

row Row id

e.Trait.1 Simulated plot error for trait 1

e.Trait.2 Simulated plot error for trait 2

`df_gv_unstr` *Genetic values example data frame*

Description

An example data frame of simulated genetic values for two traits tested in three environments. The data frame was generated using the wrapper functions unstr_asr_input and unstr_asr_output to simulate correlated genetic values based on an unstructured model for genotype-by-environment (GxE) interaction with 'AlphaSimR'.

Usage

```
df_gv_unstr
```

Format

A data frame with 700 rows and 5 columns:

env Environment id

rep Replicate number

id Genotype id

gv.Trait.1 Simulated genetic values for trait 1

gv.Trait.2 Simulated genetic values for trait 2

`field_trial_error` *Simulate plot errors in plant breeding trials*

Description

Creates a data frame with simulated plot errors for one or more traits in plant breeding trials across one or more environments. The simulated error consists of a spatial error term, a random error term and an extraneous error term. The spatial error term is constructed according to either 1) bivariate interpolation using the interp function of the package 'interp', or 2) a separable first-order autoregressive process (AR1:AR1). The random error term is constructed using an independent process. The extraneous error term is constructed as the sum of column and/or row terms. The spatial, random and extraneous error terms are combined according to a user-defined ratio.
For multiple traits, correlated error terms can be generated assuming 1) correlated spatial error between traits, 2) correlated random error between traits, 3) correlated extraneous error between traits, or 4) some combination of 1-3.
A separable covariance structure is assumed between traits and environments.

Usage

```
field_trial_error(
  n_envs,
  n_traits,
  n_reps,
  n_cols,
  n_rows,
  plot_length,
  plot_width,
  rep_dir = "column",
  var_R,
  S_cor_R = NULL,
  R_cor_R = NULL,
  E_cor_R = NULL,
  spatial_model = "bivariate",
  complexity = 12,
  col_cor = NULL,
  row_cor = NULL,
  prop_spatial = 0.5,
  prop_ext = 0,
  ext_dir = NULL,
  return_effects = FALSE
)
```

Arguments

- `n_envs` Number of environments to be simulated (same as for `compsym_asr_input` or `unstr_asr_output`, where applicable).
- `n_traits` Number of traits to be simulated.
- `n_reps` A vector defining the number of complete replicates in each environment. If only one value is provided and `n_traits` > 1, all environments will be assigned the same number of replicates.
- `n_cols` A vector defining the total number of columns in each environment. If only one value is provided and `n_traits` > 1, all environments will be assigned the same number of columns.
- `n_rows` A vector defining the total number of rows in each environment. If only one value is provided and `n_traits` > 1, all environments will be assigned the same number of rows.
- `plot_length` A vector defining the plot length (column direction, usually longer side) in each environment. If only one value is provided and `n_traits` > 1, the plots in all environments will be assigned the same plot length.
- `plot_width` A vector defining the plot width (row direction, usually shorter side) in each environment. If only one value is provided and `n_traits` > 1, the plots in all environments will be assigned the same plot width.
- `rep_dir` A character string specifying the direction of replicate blocks. One of either "column" (side-by-side, the default) or "row" (above-and-below). `rep_dir` is ignored when `n_reps` = 1.
- `var_R` A vector of error variances for each trait by environment combination (ordered as environments within traits). If the length of `var_R` is equal to `n_traits`, all environments will be assigned the same error variance for each trait.

S_cor_R	A matrix of spatial error correlations between more than one trait. If not defined and n_traits > 1, a diagonal matrix is constructed.
R_cor_R	A matrix of random error correlations between more than one trait. If not defined and n_traits > 1, a diagonal matrix is constructed.
E_cor_R	A matrix of extraneous error correlations between more than one trait. If not defined and n_traits > 1, a diagonal matrix is constructed. The same correlation is assigned to the column and row errors.
spatial_model	A character string specifying the model used to simulate the two-dimensional spatial error term. One of either "Bivariate" (bivariate interpolation, the default) or "AR1:AR1" (separable first-order autoregressive process).
complexity	A scalar defining the complexity of the bivariate interpolation model. By default, complexity = 12. Note that low values may lead to convergence problems. See interp for further details.
col_cor	A vector of column autocorrelations for each environment used in the AR1:AR1 spatial error model. If only one value is provided, all environments will be assigned the same column autocorrelation.
row_cor	A vector of row autocorrelations for each environment used in the AR1:AR1 spatial error model. If only one value is provided, all environments will be assigned the same row autocorrelation.
prop_spatial	A vector defining the proportion of spatial error variance to total error variance (spatial + random + extraneous) for each trait by environment combination. If the length of prop_spatial is equal to n_envs, all traits will be assigned the same proportion for each environment. By default, the spatial error variance accounts for half of the total error variance (prop_spatial = 0.5).
prop_ext	A vector defining the proportion of extraneous error variance to total error variance (spatial + random + extraneous) for each trait by environment combination. If the length of prop_ext is equal to n_envs, all traits will be assigned the same same proportion for each environment. By default, the extraneous error variance is zero (prop_ext = 0).
ext_dir	A character string specifying the direction of extraneous variation. One of either "column", "row" or "both". When "both", half the variance is assigned to the columns and half is assigned to the rows.
return_effects	When TRUE, a list is returned with additional entries for each trait containing the spatial, random and extraneous errors. By default, return_effects = FALSE.

Value

A data frame containing the environment, block, column and row identifiers, as well as the simulated error for each trait. When return_effects = TRUE, a list is returned with additional entries for each trait containing the spatial and random error values.

Examples

```
# Simulation of plot errors for two traits in three environments using a bivariate
# interpolation model for spatial variation.

n_envs <- 3 # Number of simulated environments.
n_traits <- 2 # Number of simulated traits.

# Field layout
n_cols <- 10 # Total number of columns in each environment.
```

```
n_rows <- c(20, 30, 20) # Total number of rows in each environment.
plot_length <- 5 # Plot length set to 5 meters in each environment.
plot_width <- 2 # Plot width set to 2 meters in each environment.
n_reps <- c(2, 3, 2) # Number of complete replicates (blocks) per environment.

# Error variances for traits 1 and 2.
var_R <- c(0.4, 15)

# Spatial error correlations between traits 1 and 2.
S_cor_R <- matrix(
  c(
    1.0, 0.2,
    0.2, 1.0
  ),
  ncol = 2
)

error_df <- field_trial_error(
  n_envs = n_envs,
  n_traits = n_traits,
  n_cols = n_cols,
  n_rows = n_rows,
  plot_length = plot_length,
  plot_width = plot_width,
  n_reps = n_reps,
  rep_dir = "row",
  var_R = var_R,
  S_cor_R = S_cor_R,
  spatial_model = "Bivariate",
  complexity = 14,
  prop_spatial = 0.6,
  prop_ext = 0.1,
  ext_dir = "row",
  return_effects = TRUE
)
```

`make_phenotypes` *Phenotype simulation through combination of genetic values and plot errors*

Description

Creates a data frame of simulated field trial phenotypes through combination of the genetic values and the plot errors generated for one or more traits with 'FieldSimR'. If the genetic values were obtained externally, they have to be arranged in a data frame with columns "env", "rep", and "id" additional to the genetic values for each trait.

Usage

```
make_phenotypes(gv_df, error_df, randomise = FALSE)
```

Arguments

`gv_df`	A data frame of genetic values. Must contain the columns "env", "rep", and "id" additional to the genetic values for each trait.

`error_df` A data frame of plot errors. Must contain the columns "env", "block", "col", and "row" additional to the error values for each trait.

`randomise` When TRUE, genotypes are randomly allocated to plots within blocks to simulate a randomized complete block design (RCBD).
Note: other experimental designs must be generated externally.

Value

A data-frame containing the environment id, block id, column id, row id, genotype id, and the phenotypic values for each trait.

Examples

```
# Create data frame of phenotypes through combination of the the genetic values and the
# plot errors from the two example data frames 'df_gv_unstr' and 'df_error_bivar'.

gv_df <- df_gv_unstr
error_df <- df_error_bivar

pheno_df <- make_phenotypes(
  gv_df,
  error_df,
  randomise = TRUE
)
```

`plot_effects` *Graphics for plot effects*

Description

Graphically displays plot effects (e.g., phenotypic values, genetic values, errors) onto a field array, where the colour gradient ranges from red (low value) to green (high value).
This function requires a data frame generated with field_trial_error as an input, or any data frame with columns named "env", "col", "row", and the effect to be displayed. If the data frame contains a column named "block", then block borders will distinguish the blocks if `blocks = TRUE`.

Usage

```
plot_effects(df, env, effect, blocks = TRUE)
```

Arguments

`df` A data frame containing the columns "env", "row", "col", and the effect to be plotted. If `df` contains a column named "block", then block borders will distinguish the blocks if `blocks = TRUE`. If `df` is a list, only the first entry will be used unless otherwise specified.

`env` The name of the environment to be plotted.

`effect` The name of the effect to be plotted.

`blocks` When TRUE (default), blocks are distinguished with block borders.

Value

Graphic of the field array, where the colour gradient ranges from red (low value) to green (high value) of the effect

Examples

```
# Plot the simulated error for trait 2 in environment 2 of the field trial error example data
# frame 'df_error_bivar'.

error_df <- df_error_bivar

plot_effects(
  error_df,
  env = 2,
  effect = "e.Trait.2"
)
```

`rand_cor_mat` *Random correlation matrix*

Description

Creates a general p x p correlation matrix with user-defined maximum and minimum correlations.

Usage

```
rand_cor_mat(p, min_cor = -1, max_cor = 1, pos_def = FALSE)
```

Arguments

`p`	A scalar defining the dimensions of the correlation matrix.
`min_cor`	A scalar defining the minimum correlation. By default, min_cor = -1.
`max_cor`	A scalar defining the maximum correlation. By default, max_cor = 1.
`pos_def`	When TRUE, the function bend of the package 'mbend' is used with default arguments to bend a symmetric non-positive-definite correlation matrix to a positive-definite matrix. By default, pos_def = FALSE.

Value

A p x p correlation matrix.

Examples

```
cor_A <- rand_cor_mat(10, min_cor = -0.2, max_cor = 0.8, pos_def = TRUE)
```

`unstr_asr_input` *Genetic values based on an unstructured model for GxE interaction using 'AlphaSimR' - Input parameters*

Description

Creates a list of simulation parameters for 'AlphaSimR' to simulate genetic values for multiple traits in multiple environments based on an unstructured model for genotype-by-environment (GxE) interaction.

By default, 'AlphaSimR' does not support complex models for GxE interaction. However, its functionality to simulate correlated genetic values can be utilised for this purpose by providing the required variance structures. `unstr_asr_input` is a wrapper function to construct the variance structures required to simulate GxE interaction in 'AlphaSimR' based on an unstructured model. This function is also used in combination with the wrapper function compsym_asr_output.

Usage

```
unstr_asr_input(
  n_envs,
  n_traits,
  mean,
  var = NULL,
  T_var = NULL,
  E_var = NULL,
  cor_A = NULL,
  E_cor_A = NULL,
  T_cor_A = NULL,
  mean_DD = NULL,
  var_DD = NULL,
  E_var_DD = NULL,
  T_var_DD = NULL,
  cor_DD = NULL,
  E_cor_DD = NULL,
  T_cor_DD = NULL,
  rel_AA = NULL,
  E_rel_AA = NULL,
  T_rel_AA = NULL,
  cor_AA = NULL,
  E_cor_AA = NULL,
  T_cor_AA = NULL
)
```

Arguments

`n_envs`	Number of environments to be simulated. A minimum of two environments is required.
`n_traits`	Number of traits to be simulated.
`mean`	A vector of mean genetic values for each trait-by-environment combination (ordered as environments within traits). Simulated traits can have a different mean for each environment. If the length of `mean` is equal to to `n_traits`, all traits will be assigned the same mean for each environment.

var
: A vector of genetic variances for each trait-by-environment combination (ordered as environments within traits). If the length of var is equal to n_traits, all traits will be assigned the same variance in each environment.
Alternatively, if a separable structure between traits and environments is desired, T_var and E_var can be provided. By default, var = NULL.

T_var
: A vector of genetic variances for each trait. Must be provided in combination with E_var.
Alternatively, var can be provided. By default, T_var = NULL.

E_var
: A vector of genetic variances for each environment. Must be provided in combination with T_var.
Alternatively, var can be provided. By default, E_var = NULL.

cor_A
: A matrix of additive genetic correlations between all trait-by-environment combinations. If not defined and n_traits > 1, a diagonal matrix is constructed.
Alternatively, T_cor_A and E_cor_A can be provided.

E_cor_A
: A matrix of additive genetic correlations between more than one environment. Must be provided in combination with T_cor_A.
Alternatively, cor_A can be provided. By default, E_cor_A = NULL.

T_cor_A
: A matrix of additive genetic correlations between more than one trait. Must be provided in combination with E_cor_A.
Alternatively, cor_A can be provided. By default, T_cor_A = NULL.

mean_DD
: A vector of mean dominance degrees for each trait-by-environment combination (ordered as environments within traits), similar to mean. If the length of mean_DD is equal to n_traits, all traits will be assigned the same mean_DD for each environment. By default, mean_DD = NULL and dominance is not simulated.

var_DD
: A vector of dominance degree variances for each trait-by-environment combination (ordered as environments within traits), similar to var. If the length of var_DD is equal to to n_traits, all traits will be assigned the same var_DD for each environment.
Alternatively, if a separable structure between traits and environments is desired, T_var_DD and E_var_DD can be provided. By default, var_DD = NULL.

E_var_DD
: A vector of dominance degree genetic variances for each environment, similar to E_var. Must be provided in combination with T_var_DD.
Alternatively, var_DD can be provided. By default, E_var_DD = NULL.

T_var_DD
: A vector of dominance degree variances for each trait, similar to T_var. Must be provided in combination with E_var_DD.
Alternatively, var_DD can be provided. By default, T_var_DD = NULL.

cor_DD
: A matrix of dominance degree correlations between all trait-by-environment combinations, similar to cor_A. If not defined and n_traits > 1, a diagonal matrix is constructed.
Alternatively, T_cor_DD and E_cor_DD can be provided. By default, cor_DD = NULL.

E_cor_DD
: A matrix of dominance degree correlations between more than one environment, similar to E_cor_A. Must be provided in combination with T_cor_DD.
Alternatively, cor_DD can be provided. By default, E_cor_DD = NULL.

T_cor_DD
: A matrix of dominance degree correlations between more than one trait, similar to T_cor_A. Must be provided in combination with E_cor_DD.
Alternatively, cor_DD can be provided. By default, T_cor_DD = NULL.

`rel_AA` A vector defining the magnitude of additive-by-additive (epistatic) variance relative to additive genetic variance for each trait-by-environment combination (ordered as environments within traits), that is in a diploid organism with allele frequency 0.5. If the length of `rel_AA` is equal to to `n_traits`, all traits will be assigned the same rel_AA for each environment.
Alternatively, if a separable structure between traits and environments is desired, `T_rel_AA` and `E_rel_AA` can be provided. By default, `rel_AA = NULL` and epistasis is not simulated.

`E_rel_AA` A vector defining the magnitude of additive-by-additive (epistatic) variance relative to the additive genetic variance for each environment, that is in a diploid organism with allele frequency 0.5. Must be provided in combination with `T_rel_AA`.
Alternatively, `rel_AA` can be provided. By default, `E_rel_AA = NULL`.

`T_rel_AA` A vector defining the magnitude of additive-by-additive (epistatic) variance relative to the additive genetic variance for each trait, that is in a diploid organism with allele frequency 0.5. Must be provided in combination with `E_rel_AA`.
Alternatively, `rel_AA` can be provided. By default, `T_rel_AA = NULL`.

`cor_AA` A matrix of epistatic correlations between all trait-by-environment combinations, similar to `cor_A`. If not defined and `n_traits > 1`, a diagonal matrix is constructed.
Alternatively, `T_cor_AA` and `E_cor_AA` can be provided. By default, `cor_AA = NULL`.

`E_cor_AA` A matrix of epistatic correlations between more than one environment, similar to `E_cor_A`. Must be provided in combination with `T_cor_AA`.
Alternatively, `cor_AA` can be provided. By default, `E_cor_AA = NULL`.

`T_cor_AA` A matrix of epistatic correlations between more than one trait, similar to `T_cor_A`. Must be provided in combination with `E_cor_AA`.
Alternatively, `cor_AA` can be provided. By default, `T_cor_AA = NULL`.

Details

`unstr_asr_input` can handle non-separable and separable structures between traits and environments.

- For non-separable structures, provide (1) `var`, and (2) `cor_A`.
- For separable structures, provide (1) `T_var` & `E_var`, and (2) `T_cor_A` & `E_cor_A`.

Note: 'AlphaSimR' can simulate different biological effects (see: `SimParam`).

- For additive traits use `addTraitA()`.
- For additive + dominance traits use `addTraitAD()`.
- For additive + epistatic traits use `addTraitAE()`.
- For additive + dominance + epistatic traits use `addTraitADE()`.

If non-additive effects are to be simulated, check the `useVarA` argument of these functions.

Value

A list containing input parameters for 'AlphaSimR', which is used to simulate correlated genetic effects based on an unstructured model.

Examples

```
# Simulation of genetic values in 'AlphaSimR' for two additive + dominance traits tested in
# three environments based on an unstructured model for GxE interaction.

# 1. Define the genetic architecture of the simulated traits.
# Mean genetic values and mean dominance degrees for trait 1 in all 3 environments and trait 2
# in all 3 environments.
mean <- c(1, 3, 2, 80, 70, 100) # Trait 1 x 3 environments, trait 2 x 3 environments.
mean_DD <- c(0.1, 0.4) # Trait 1 and 2, same values set in all 3 environments for each trait.

# Additive genetic variances (useVarA = TRUE) and dominance degree variances for traits 1 and 2,
# assuming a separable structure between traits and environments.
T_var <- c(0.2, 10) # Genetic variances defined for the two traits.
E_var <- c(0.5, 1, 1.5) # Genetic variances defined for the three environments.

# Dominance degree variances for trait 1 in 3 environments and for trait 2 in 3 environments,
# assuming a non-separable structure between traits and environments.
var_DD <- c(0.1, 0.15, 0.2, 0.2, 0.3, 0.2)

# Additive genetic correlations between the two simulated traits.
T_cor_A <- matrix(
  c(
    1.0, 0.3,
    0.3, 1.0
  ),
  ncol = 2
)

# Additive genetic correlations between the three simulated environments.
E_cor_A <- stats::cov2cor(matrix(
  c(
    0.5, 0.4, 0.6,
    0.4, 1.0, 0.5,
    0.6, 0.5, 1.5
  ),
  ncol = 3
))

# Dominance degree correlation between all six trait-by-environment combinations.
cor_DD <- diag(6) # Assuming independence between traits

input_asr <- unstr_asr_input(
  n_envs = 3,
  n_traits = 2,
  mean = mean,
  T_var = T_var,
  E_var = E_var,
  T_cor_A = T_cor_A,
  E_cor_A = E_cor_A,
  mean_DD = mean_DD,
  var_DD = var_DD,
  cor_DD = cor_DD
)
```

unstr_asr_output *Genetic values based on an unstructured model for GxE interaction using 'AlphaSimR' - Simulated genetic values*

Description

Creates a data frame of correlated genetic values for multiple traits in multiple environments based on an unstructured model for genotype-by-environment (GxE) interaction. This function requires an 'AlphaSimR' population object generated using the unstr_asr_input function.

Usage

```
unstr_asr_output(pop, n_envs, n_reps, n_traits)
```

Arguments

pop	An 'AlphaSimR' population object (`Pop-class` or `HybridPop-class`) generated using unstr_asr_input.
n_envs	Number of simulated environments (same as in unstr_asr_input).
n_reps	A vector defining the number of complete replicates in each environment. If only one value is provided and `n_traits` > 1, all environments will be assigned the same number of replicates.
n_traits	Number of simulated traits (same as in unstr_asr_input).

Value

A data frame containing the environment id, replicate number, genotype id, and the simulated genetic values for each trait.

Examples

```
# Simulation of genetic values in 'AlphaSimR' for two additive + dominance traits tested in
# three environments based on an unstructured model for GxE interaction.

# 1. Define the genetic architecture of the simulated traits.
# Mean genetic values and mean dominance degrees for trait 1 in all 3 environments and trait 2
# in all 3 environments.
mean <- c(1, 3, 2, 80, 70, 100) # Trait 1 x 3 environments, trait 2 x 3 environments.
mean_DD <- c(0.1, 0.4) # Trait 1 and 2, same values set in all 3 environments for each trait.

# Additive genetic variances (useVarA = TRUE) and dominance degree variances for traits 1 and 2,
# assuming a separable structure between traits and environments.
T_var <- c(0.2, 10) # Genetic variances defined for the two traits.
E_var <- c(0.5, 1, 1.5) # Genetic variances defined for the three environments.

# Dominance degree variances for trait 1 in 3 environments and for trait 2 in 3 environments,
# assuming a non-separable structure between traits and environments.
var_DD <- c(0.1, 0.15, 0.2, 0.2, 0.3, 0.2)

# Additive genetic correlations between the two simulated traits.
T_cor_A <- matrix(
  c(
```

```
    1.0, 0.3,
    0.3, 1.0
  ),
  ncol = 2
)

# Additive genetic correlations between the three simulated environments.
E_cor_A <- stats::cov2cor(matrix(
  c(
    0.5, 0.4, 0.6,
    0.4, 1.0, 0.5,
    0.6, 0.5, 1.5
  ),
  ncol = 3
))

# Dominance degree correlation between all six trait-by-environment combinations.
cor_DD <- diag(6) # Assuming independence between traits

input_asr <- unstr_asr_input(
  n_envs = 3,
  n_traits = 2,
  mean = mean,
  T_var = T_var,
  E_var = E_var,
  T_cor_A = T_cor_A,
  E_cor_A = E_cor_A,
  mean_DD = mean_DD,
  var_DD = var_DD,
  cor_DD = cor_DD
)


# 2. Use input_asr to simulate genetic values in 'AlphaSimR' based on an unstructured model for
# GxE interaction.

library("AlphaSimR")
FOUNDERPOP <- quickHaplo(
  nInd = 100,
  nChr = 6,
  segSites = 100
)

SP <- SimParam$new(FOUNDERPOP)

SP$addTraitAD(
  nQtlPerChr = 100,
  mean = input_asr$mean,
  var = input_asr$var,
  meanDD = input_asr$mean_DD,
  varDD = input_asr$var_DD,
  corA = input_asr$cor_A,
  corDD = input_asr$cor_DD,
  useVarA = TRUE
)

# By default, the value provided in 'var' represents the additive variance.
```

```
# If useVarA=FALSE, 'var' represents the total genetic variance.

pop <- newPop(FOUNDERPOP)


# 3. Create a data frame containing the simulated genetic values for the two traits in the
# three environments.

n_reps <- c(2, 3, 2) # Vector containing the number of complete replicates in each environment.

gv_df <- unstr_asr_output(
  pop = pop,
  n_envs = 3,
  n_reps = n_reps,
  n_traits = 2
)
```